BETTER PROGRAMMING FOR YOUR
SPECTRUM AND ZX81

S. ROBERT SPEEL
Consultant editor: TIM HARTNELL

OVER 40 NEW PROGRAMS

Robert Speel was born in 1964. He was one
of the first people to acquire a ZX80. He started
writing software immediately and moved on
to the ZX81 and Spectrum as soon as they were
introduced. He is a keen player of chess and
fantasy role-playing games and is also
interested in natural history.

Tim Hartnell is editor of *ZX Computing*
magazine and co-ordinates the National
ZX Users Club. He is the author of several
computer books including *The ZX Spectrum
Explored* (1982).

S. Robert Speel

Better Programming for Your Spectrum and ZX81

The programs presented in this book have been
included for their instructional value. They have
been tested with care but they are not guaranteed
for any particular purpose. While every care has
been taken the publishers cannot be held
responsible for running mistakes that may
occur.

Contents

Foreword
by Tim Hartnell

Welcome to this book on programming with the ZX computer. Small in size, the performance of the Spectrum and ZX81 can be most impressive, and this book by Robert Speel is the key to proving just how impressive the Sinclair computers can be.

The author and I held many discussions while the book was being written, as we worked out the topics which were most likely to be of interest and benefit to programmers. At first we concentrated entirely on games programs, then realized that this could needlessly limit the value and scope of the book. Therefore, although the emphasis is on games programs, and games programming, we have several more serious programs to show how the Spectrum and ZX81 can earn their keep. However, the games techniques will serve you well, no matter which type of programs you eventually write.

Such programs as Hero Maker and Knight Fight are as good as much of the better commercial software available, so you are in for a great deal of enjoyment when you enter and run the programs. However, the value of the book would be diminished if you saw it only as a collection of programs to type in and run on your computer. By all means use the programs just as they are at first, but then make them departure points for your own program development and customizing. This way, you'll put your own stamp on the programs, and will develop your programming skills.

The book contains the key to better programming on your computer. It is up to you to turn it.

London, 1982

Introduction

If you have read your user manual and want to improve your collection of programs and your programming skills, this is the book for you. I have included many programs which are different from those found elsewhere. Each program is accompanied by an explanation of listing, which tells you what each section of the program does and discusses programming techniques used.

I have started off by looking at Spectrum colour and sound, which can enhance many programs considerably. Palette, Lightning and First Contact are among the programs which highlight these features.

Then there are chapters on user-defined characters, which are among the Spectrum's most powerful features, and I have included some programs for these. High-resolution graphics are next, with some short programs which show their use. There is a section giving more general information, including a series of modules to move objects under user control, and a discussion on ways of avoiding problems with programs. You will find a number of suggestions to help you debug your programs with minimum fuss.

A series of longer programs follows. My 3-D Maze uses high-res graphics to draw your view of the path ahead: you must escape from the maze as quickly as possible – a 'help' option is included for the frustrated. In Boat Race, you can bet on one of four small user-defined craft in an attempt to win a fortune. Sheepdog puts you in the dog's position, trying to herd four sheep into a neighbouring field. A veteran sheepdog can attempt six sheep, which need chasing through a sheep-dip too! The next program, Knight Fight, places you on horseback, jousting against a fierce opponent: when one contestant is

unseated the battle continues on foot and you use your chosen weapon to defeat your enemy. In Space Raider you command a small spaceship under attack by three different types of alien, each with its own way of getting you.

A couple of serious programs are next. Probability Check takes a look at random events. Wave Addition shows destructive and constructive interference using multi-coloured high-res graphics.

To help you use the many ZX81 programs in this book and elsewhere, I have included a section on converting programs to the Spectrum.

The Patterns section shows ways of making effective use of low-resolution graphics as found on the ZX81, and a powerful drawing program – Screen Art – lets you design your masterpieces directly on the screen.

Text Display and Avoid are two of the shorter programs in the book, fitting in 1K, and use sideways and vertical scrolling respectively.

The next program, Hero Maker, is one of my longest programs, and one of the most important in the book. If you are a fan of fantasy role-playing games you will understand this game very quickly. You will be pleased to see that, unlike many computer Adventure games, this program runs differently every time and you will face fierce monsters guarding treasure, weapons and life-giving water.

For a more peaceful game, you can play a relaxing round of golf in Fairway, a graphically illustrated course. To sate your desire for space games, Alien Descender makes you, for once, the alien, sinking down a narrow gorge in the sea depths, avoiding the walls and the occasional depth charge. Asteroid Belt puts you at the outskirts of an asteroid swarm and if you survive at first, the asteroids get bigger and more frequent.

To put the ZX81 in a perplexing situation, in Amaze you draw the maze for the computer to solve.

Rabbits is a semi-serious program in which you are trying to breed valuable rabbits with rare characteristics. They breed, of course, according to Mendel's laws of genetics, passing their unwanted blotches to their multitude of descendants.

The programs now turn completely serious. Base conversions, to and from base 10, are useful 1K programs. Quadratic Equation Solver and Right-angled Triangle Solver not only give answers to problems, but also explain the details of the methods used, showing all working-out.

Compound Interest shows how your money can grow at a surprising rate from a modest outlay. With Bar Graph you can get a graphic idea of your monthly profits, perhaps from your cunning use of Compound Interest. Conversion rounds off the programs, and will convert centigrade to fahrenheit, radians to degrees; in fact any values in any steps if you know the conversion formula, printing out its results in a neat list which can be copied on the ZX Printer.

Finally, I have discussed ways of making a cassette collection, and how to try to solve hardware problems. Other useful information includes a glossary of terms used and report codes.

As you begin to adapt, enlarge and transform my programs into your own, I think you will find the explanations of listings very useful in making you a better programmer.

Best of programming,

S. Robert Speel
Eastcote, 1982

Note

Later models of the ZX81 and all models of the Spectrum call the NEWLINE key ENTER. If your computer has an ENTER key, all references to NEWLINE refer to that key.

Spectrum Colour

The ZX Spectrum's name is actually derived from its colour capabilities and there are several commands dealing with colour. There are some very powerful features, such as the fact that all 8 colours can be used at once, with both text and hi-res graphics, and the ease of using the commands.

Here is a short summary of the colour commands. For all the colour commands, codes are used for each colour. These are:

    0 Black
    1 Blue
    2 Red
    3 Magenta (purple)
    4 Green
    5 Cyan (pale blue)
    6 Yellow
    7 White

There are 32 columns and 22 lines of characters on the upper (easily accessible) part of the screen – 704 character positions in all. Each of these has a foreground colour INK, and a background colour PAPER. Each position has two possible levels of brightness and can be normal or flashing. Together, the INK, PAPER, BRIGHT and FLASH numbers are called ATTRibutes of a character. All the commands can be set globally – i.e., so that all PRINTing and PLOTting after the command will have that ATTRibute. This is done by a simple line without PRINT or PLOT. To make the ATTRibute come into effect all over the screen, CLS is used. Thus to set red INK on blue PAPER, flashing and bright, you could use

100 INK 2: PAPER 1: FLASH 1: BRIGHT 1: CLS

The colour commands can be set for just one PRINT or PLOT command by placing the colour command after the PRINT or PLOT, followed by a semicolon – e.g.:

  PRINT INK 2; PAPER 7; FLASH 1; "Hi there"

If you are using PRINT AT, colour commands go *before* the AT. In fact, you can change within one PRINT statement several times:

  PRINT INK 7; PAPER 1; "HELLO"; FLASH 1; A$;
  FLASH 0; INVERSE 1; AT 10,4; "Press any key"; INK 4;
  PAPER 0; AT 15,12; B$

  BRIGHT and FLASH can be set to 0 – off, or 1 – on, although whether a character is BRIGHT or not is often very difficult to tell.
  INVERSE and OVER can also be set to 0 or 1.
  INVERSE 1 swaps the dot pattern in a character, so 1s (dots) become noughts, and vice versa. Note that this is *not* the same as swapping the INK and PAPER colours (although it appears the same on screen) and these remain unaltered.
  OVER 1 adds any new characters on top of the old ones on the screen. So if either has an INK colour at a point, the resulting character will have an ink spot. But if both dots are ink colour, the result is paper colour. This is an XOR (Exclusive OR) function, giving ink if the first character has ink at a dot position or the second has ink there, but not both. INVERSE and OVER are used in the same way as FLASH and BRIGHT.
  Two other commands, INVERSE VIDEO and TRUE VIDEO, are treated rather vaguely in the manual. They are more comparable to the graphics key than any other commands, as there is no indicator in the listing except the actual 'inversed' or graphic characters.
  You can use INVERSE VIDEO to PRINT white on black inside the quotes of PRINT statements.
  If you press INVERSE VIDEO, followed by GRAPHICS,

all the graphics are reversed, so the graphics you normally get by pressing a number key are then produced with CAPS SHIFT and the number key. To get back to normal mode from INVERSE video mode press TRUE VIDEO.

Note that the cursor is often reluctant to go over inverse characters, and you may have to persevere strongly. You can, just by EDITing a line and pressing INVERSE VIDEO, get the whole line (except the line number) in white on black. When you put the line back by pressing ENTER the whole of the listing after that point will change to INVERSE VIDEO. This will give you permanent inverse listings, including on the ZX Printer. However, PRINT statements in the inversed lines, even though their contents *look* inversed, will PRINT on screen in normal mode (unless separately inversed). You will only fully understand these two commands with practice.

Here are a few programs which use some of the above commands to produce interesting and very colourful displays.

Colour Spectrum

To use
Press RUN and watch.

The program produces a highly colourful display showing all the Spectrum colours in squares. It changes these by considering the ATTRibutes of each character square and adding 1.

Notes on listing
10 PRINT lines of colour.
20 If paper colour <7 then goto 40, and add 1.
30 Change paper colour to 0.
50 Repeat for random number of lines.
60 Necessary as lines 40 and 50 used by the second part of the program as well as the first.
70–80 Same as first part of program, but starting at bottom of screen instead of top.

Listing

```
   1 REM Colour Spectrum

  10 RANDOMIZE : FOR f=1 TO 80:
FOR g=0 TO 7: PRINT PAPER g;" ";
: NEXT g: NEXT f

  20 LET Z=1: FOR f=0 TO INT (RN
D*22): FOR g=0 TO 31: IF ATTR (f
,g) <56 THEN GO TO 40
  30 PRINT AT f,g; PAPER 0;" ":
GO TO 50
  40 PRINT AT f,g; PAPER (ATTR (
f,g)/8+1);" "
  50 NEXT g: NEXT f

  60 IF z=2 THEN GO TO 20
  70 LET z=2: FOR f=21 TO INT (R
ND*22) STEP -1: FOR g=31 TO 0 ST
EP -1: IF ATTR (f,g) <56 THEN GO
TO 40
  80 PRINT AT f,g, PAPER 0;" ":
GO TO 50
```

Stars and Stripes

To use

Just press RUN and wait for a few seconds.

This program DRAWs the star pattern in just four lines. The rest of the program PRINTs the stripes and occasionally swaps them round. Since the stripes are made of coloured squares, they do not show at all on the COPYed picture.

Notes on listing

10 The effect of this line is that the DRAWn pattern gradually appears as the program runs.

Note that if you have a printer you can STOP the program after line 70, and COPY a seemingly blank screen to get the above pictures.

20–70 DRAW picture and PRINT ZX SPECTRUM.

Note that the same DATA is used to DRAW all the stars.

100–140 Colouring routine. Gradually the paper colour is

changed, and the ink colour is altered to contrast. Every now and again the colours are swapped round and the routine repeats.

500 DATA for star shape.
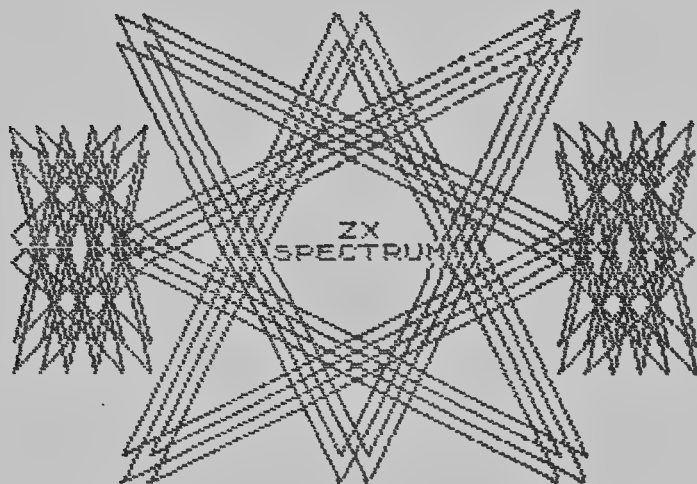
Listing

```
   1 REM Stars and Stripes
   2 REM  © S.Robert Speel

  10 INK 7: PAPER 7: CLS
  20 FOR f=0 TO 10 STEP 10: FOR
g=0 TO 10 STEP 10: RESTORE : PLO
T 40+f,g: FOR h=1 TO 8: READ a,b
: DRAW a*80,b*80: NEXT h: NEXT g
: NEXT f
  30 FOR f=0 TO 10 STEP 10: FOR
g=0 TO 10 STEP 10: FOR h=0 TO 20
0 STEP 200: RESTORE
  40 PLOT f+h,g+40: FOR i=1 TO 8
: READ a,b: DRAW a*20,b*40: NEXT
 i: NEXT h: NEXT g: NEXT f
  70 PRINT AT 10,15;"ZX";AT 11,1
2;"SPECTRUM"
```

```
  100 FOR f=1 TO 300: LET a=INT (
RND*20): LET b=INT (RND*8)*4
  110 PRINT PAPER (b/4); INK 9; O
VER 1;AT a,b;"      ";AT a+1,b;"
   ";AT a+2,b;"      ": NEXT f
  120 FOR f=1 TO 300: LET a=INT (
RND*20): LET b=INT (RND*8)*4
  130 PRINT PAPER (7-b/4); INK 9;
OVER 1;AT a,b;"      ";AT a+1,b;"
    ";AT a+2,b;"      ": NEXT f
  140 GO TO 100

  500 DATA 1,2,1,-2,-2,1,2,1,-1,-
2,-1,2,2,-1,-2,-1
```

Doorway

To use
RUN and watch.

This program uses FLASH and OVER to achieve some very
eye-catching colour effects.
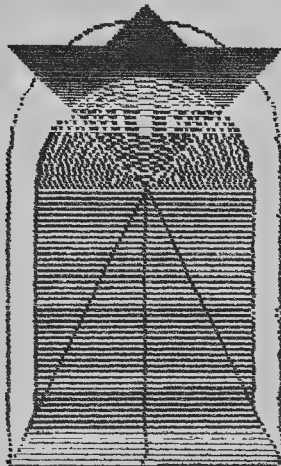
Notes on listing
20–30 DRAW lower triangle and fill in with horizontal lines.

40 DRAW upside-down triangle and fill in solid.
50 DRAW small 'peak' triangle and fill in solid.
60 DRAW archway.
70 DRAW horizontal lines in lower part of door.
80 Fill in top of archway by DRAWing semicircles.
90 Central vertical line.
100 Colouring. The border is constantly changed and the g and h loops move down the door PRINTing in different colours giving a cascading effect. Note that the many horizontal lines prevent the flashing from being too dominant.
Note what happens when the program stops.

Listing

```
   1 REM Doorway
   2 REM  © S.Robert Speel

  10 INK 1: PAPER 6: BORDER 3: B
RIGHT 1: CLS
  20 PLOT 78,0: DRAW 50,100: DRA
W 50,-100
  30 FOR f=100 TO 1 STEP -2: PLO
T 128-f/2,100-f: DRAW f,0: NEXT
f
  40 FOR f=100 TO 1 STEP -1: PLO
T 128-f/2,100+f/2: DRAW f,0: NEX
T f
  50 FOR f=0 TO 20: PLOT 128-f,1
65-f: DRAW f*2,0: NEXT f

  60 FOR f=0 TO 10 STEP 10: PLOT
 88-f,20-f*2: DRAW 0,80+f*3: DRA
W 80+f*2,0,-PI: DRAW 0,-80-f*3:
NEXT f
  70 FOR f=20 TO 100 STEP 2: PLO
T 88,f: DRAW INK 0;80,0: NEXT f
  80 FOR f=1 TO 40 STEP 2: PLOT
OVER 1;128-f,100: DRAW INK 2; OV
ER 1;2*f,0,-PI: NEXT f
  90 PLOT 128,100: DRAW 0,-100

 100 FOR f=1 TO 50: FOR g=9 TO 2
0 STEP 2: INK RND*8: FOR h=0 TO
1: BORDER RND*7: PRINT PAPER RND
*8; OVER 1; FLASH 1;AT g+h,11;"
            ": NEXT h: NEXT g: NEXT
f
```

Flashclash

This program shows how much you can cram into one program line. I had to put the REM statement at the end, as everything after the REM is treated as part of the REM, including the colon.

The program uses FLASH and all 8 colours to great effect, but leaves some parts of the screen stable to give some relief to the eye.

To use
RUN and watch.

Try altering the inverse spaces to normal spaces, or remove the FLASH to get a totally different effect.

Note that the pattern COPYed using the ZX Printer is not apparent on the screen at all.

Listing

```
  10 FOR e=0 TO 5: FOR f=0 TO 9:
 FOR g=f TO 20-f STEP (2+e): FOR
 h=f TO 30-f STEP (2+e): PRINT F
LASH INT (RND*2); INK f; PAPER A
BS (7-f);AT g,h;"█"; INK ABS (7-
f); PAPER f;" ";AT g+1,h;"█"; IN
K f; PAPER ABS (7-f);" ": NEXT h
: NEXT g: NEXT f: NEXT e: REM Fl
ashclash © 1982 ,S.Robert Speel
```

Palette

This program allows you to get lots of extra colours from your Spectrum. With a 'chessboard' user-defined graphic, we can blend ink and paper colours to produce many new colours. This program goes through the whole range.

To use
RUN to give the first colour – black. The ink and paper colours

are given. Press ENTER to get the next shade. Repeat the cycle through full range.

Notes on listing

10 Set up user-defined 'chessboard' pattern.
20 For all 8 paper colours, 8 ink colours, and for each of 21 lines.
30 Print line of chessboard graphics. Note these are graphic As, *not* normal capital letters.
40 Print ink and paper colours, instructions to press ENTER.
50 Repeat when key is pressed for all combinations of ink and paper.

Listing

```
   1 REM   palette
   2 REM   © S.Robert Speel 1982

  10 FOR f=0 TO 7 STEP 2: POKE U
SR CHR$ 144+f,170: POKE USR CHR$
 144+f+1,85: NEXT f
  20 FOR f=0 TO 7: FOR g=0 TO 7:
FOR h=0 TO 20
  30 PRINT PAPER f; INK g;"AAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  40 NEXT h: PRINT AT 10,9; INK
0; PAPER 7;"INK ";g;" PAPER ";f;
AT 21,0;"press ENTER for next co
lour"
  50 PRINT AT 0,0;: PAUSE 0: NEX
T g: NEXT f
```

2×Res Colour Patterns

This program uses a special user-defined graphic divided into four parts, one all ink, one all paper, and two intermediate shades. This means that using two colours – one ink, one paper – we can get four different shades in one character square, hence the name of the program.

To use
RUN and watch.

The program goes through all the colour combinations and takes a long time to run completely. Be patient at first. If you cannot wait, after a few minutes of RUNning, BREAK, and type GOTO 40.

Notes on listing

10 Set user-defined graphic which is used to give four colours.

20 Fill screen with graphic.

30 Part 1 POKE random position on screen with INK and PAPER ATTRibutes.

This happens 500 times for each combination of INK and PAPER, but is worth waiting for.

40 Now we have FLASHing colours, although these are not apparent at first.

100 DATA for 4-colour graphic.

Listing

```
   1 REM 2 x res Colour Pattern
   2 REM  © S.Robert Speel 1982

  10 RANDOMIZE : FOR f=0 TO 7: R
EAD a: POKE USR CHR$ 144+f,a: NE
XT f
  20 FOR f=1 TO 22: FOR g=1 TO 3
2: PRINT CHR$ 144;: NEXT g: PRIN
T : NEXT f

  30 FOR f=0 TO 63: FOR g=1 TO 5
00: POKE 22528+INT (RND*704),f:
NEXT g: NEXT f
  40 FOR f=128 TO 192: FOR g=1 T
O 500: POKE 22528+INT (RND*704),
f: NEXT g: NEXT f
 100 DATA 240,240,240,240,10,165
,10,165
```

Spectrum Sound

Use of sound on the ZX Spectrum is very easy: the single sound command, BEEP, is followed by two numbers, the first of which is the length of note, the second the pitch of the note. However, that is it. We cannot play more than one note at a time, and the note cannot change as it is played. Another problem is that when BEEP is used, all calculations stop. This means that you cannot have continuous sound while a program is doing something on screen. There is no white noise generator (white noise is used for explosions, growls and mechanical noises) which would be useful in many games.

So what use can be made of Sinclair sounds?

Firstly, it is used as a keyboard sounder, something that many ZX81 users buy separately.

The keyboard sounder is rather quiet, so, if you want, you can make it more noticeable with a useful POKE: address 23609 holds the systems variable PIP. It is normally set at 0, but you can POKE it safely to other values. POKE 23609, 30 gives a short bleep which is louder than the usual click when a key is pressed. Try other values until you find one which suits you best. Numbers over 100 give bleeps that are too long for comfortable keyboard use.

If we consider actual programs where sound is used, the BEEP command appears in a different light. For serious programs, including business programs, scientific, maths and programming-aid programs, the only need for sound is a short note to acknowledge inputted information, and perhaps a warning sound when the computer receives a wrong bit of information. The BEEP command can easily cope with this.

Educational programs are similar in use of sound, but short trills or tunes may be used as 'rewards' when a set of questions

has been answered well, or a section of the program has been RUN successfully. Again, BEEP is suitable.

Word games need few sound effects, so the main area where better sound effects are required is in graphic-based games. Many of these would benefit greatly from use of sound, and here we must be careful.

Continuous sound while a game is being played is not possible, as the program stops while the sound is made. The best substitute we can manage is to have short BEEPs, duration 0.1 or less, in the program. The noise can be made every 'go' (i.e., each time the main loop is executed), or a special sound used when firing, hitting, or whatever special activities (activities which do not happen every go) the program has. Longer sounds can be used for explosions and when winning (or losing) a game. This way we can achieve a fairly noisy game. Remember that many games need to be slowed down to be playable, often with dummy FOR-NEXT loops. These can be replaced with BEEP commands.

In most action games, the sounds required are not musical notes. For the clashing of swords, the hum of laser beams, the roar of a dinosaur, simple beeps are no good. Using BEEP, we can actually obtain quite a few of the more exotic noises. The trick is to use the fact that the loudspeaker cannot cope with very low or short notes. We can get a large range of clicks, thuds, beats and so on, useful for engine noises, running sounds and shooting noises, by BEEPing low or very short notes as well as some of the very high notes.

If you wish to amplify these sounds, plug in the SAVE leads and press the Record button on your cassette recorder without pressing forward.

```
   10 BEEP .6,18: BEEP .7,14.5: G
O TO 10
```

        Siren

```
   10 FOR f=1 TO 50: BEEP .05,60-
f: NEXT f
```

  Fast descending scale

```
10 FOR f=1 TO 10: BEEP .01,1+f
: BEEP .01,10: NEXT f: GO TO 10
```

Firing

```
10 FOR f=1 TO 30: BEEP .05,20:
BEEP .05,0: NEXT f
```

Warning signal

```
10 FOR f=1 TO 10: BEEP .5,40:
PAUSE 7: BEEP .5,40: PAUSE 35: N
EXT f
```

Digital watch alarm

```
10 FOR f=1 TO 10: FOR g=1 TO 2
: FOR h=1 TO 8: BEEP .03,24: NEX
T h:: PAUSE 6: NEXT g: PAUSE 50:
 NEXT f
```

Telephone

```
10 BEEP .005,5: BEEP .005,-2:
PAUSE 2: GO TO 10
```

Motor

```
10 FOR f=1 TO 10 STEP 2: FOR g
=1 TO 10 STEP 2: FOR h=1 TO 10 S
TEP 2: BEEP .1,f+g-h: BEEP .1,g+
h-f: BEEP .1,h+f-g: NEXT h: NEXT
 g: NEXT f
```

Background sound
 (eg. used while reading program
        instructions)

```
10 FOR f=10 TO 20: BEEP .003,1
0: PAUSE 6-f/5: BEEP .003,5: PAU
SE 6-f/5: BEEP .003,0: PAUSE 5-f
/5: BEEP .003,1: PAUSE 25-f: NEX
T f
```

```
  20 FOR f=20 TO 10 STEP -1: BEE
P .003,10: PAUSE 6-f/5: BEEP .00
3,5: PAUSE 6-f/5: BEEP .003,0: P
AUSE 5-f/5: BEEP .003,1: PAUSE 2
5-f: NEXT f: GO TO 10

Galloping horses
```

Lightning

This program makes use of very short BEEPs to create a crackling noise, which sounds somewhat like lightning-cracks during a thunder storm. This noise is accompanied by bright flashes of 'lightning' to give quite an effective display.

To use
RUN and watch.

As the flashes occur randomly you may be faced with a blank screen for several seconds at a time.

Notes on listing
210–230 Main lightning routine. 230 selects a random stroke of lightning from the data list and DRAWs it. OVER is used to wipe it out. The random BEEP is very short to give a crackling noise.
240 Longer crackling sound, using lower notes but for a longer period.
9000–9070 DATA for various shapes of strokes of lightning.

Listing

```
  1 REM Lightning
  2 REM  © S.Robert Speel 1982

100 PAPER 0: BORDER 0: CLS : RA
NDOMIZE

200 FOR f=1 TO 10000
210 IF RND<.5 THEN PAUSE 50: GO
TO 240
```

```
 220 LET x =RND*120+50: BORDER 6:
 PAPER 6: CLS : BORDER 0: PAPER
0: CLS : LET q=9000+INT (RND*8)*
10
 230 FOR g=6 TO 0 STEP -6:: OVER
 1: INK g: PLOT x,175: RESTORE Q
: FOR h=1 TO 6: READ z1,z2: DRAW
 z1,z2: BEEP .001,RND*50: NEXT h
: NEXT g
 240 IF RND<.6 THEN NEXT f: FOR
z=1 TO 20: BEEP .01*RND,-z: NEXT
 z: IF RND<.5 THEN GO TO 240
 250 NEXT f

9000 DATA -30,-10,50,-40,-10,10,
40,-30,-90,-10,40,-40
9010 DATA 50,-40,-100,-10,10,-5,
40,-30,-50,-60,60,-20
9020 DATA 30,-10,-20,-20,40,-10,
-40,-30,10,-30,-40,-50
9030 DATA -30,-10,50,-40,-10,10,
40,-30,-90,-10,40,-40
9040 DATA 0,-50,40,-10,-80,-5,30
,-10,40,-20,-5,-40

9050 DATA -50,-5,10,-5,-5,-10,40
,-20,-10,-15,65,-70
9060 DATA -20,-40,30,30,20,-50,2
0,30,-10,-40,30,-60
9070 DATA 30,-40,-10,-20,30,10,-
20,-40,10,-10,-90,-60
```

First Contact

This program shows use of sound in several different ways. Typical 'arcade game' noises when moving your base or the alien ships are used, and a firing noise where appropriate. There is an explosion noise, and a short beep when you press a key to start a new game. There is also a siren noise for when you lose and are taken away by ambulance.

To use

It is the year 2000 and some aliens from the stars have discovered Earth. They are sending down special envoys to make contact with their brother sapients. It is your task to keep

them at bay with your missile base and protect the *status quo*. You have 20 missiles. The aliens come two at a time and when you destroy one, another appears at the top of the screen. You move left and right with the 5 and 8 keys, and fire with the 7 key. You score 1 for each alien destroyed, and when your score reaches 5, you get 0, 1 or 2 extra missiles for every new alien blasted. The game ends when an alien lands. Your score and the current highscore are given.

Notes on listing

30–40 hs = the highscore, sc = your score, sh = number of shots left, t is set when you fire, and gives your current score and missile count.

50 PRINT your base.

100 Array x: contains the enemy spaceships' co-ords, and their movement vectors. a is the x co-ord of your missile base.

200–410 This is the main routine. You may wonder why one FOR-NEXT loop is not used, rather than 3 exactly similar loops at line 200, 200–400, and 400–410. This is because if the aliens are PRINTed separately, you get this sequence of events: (1) PRINT alien 1; (2) You shoot/move; (3) Move alien 1; (4) PRINT alien 2; (5) You shoot/move; (6) Move alien 2, goto step 1. This is fine unless in 5 you shoot at alien 1. If alien 1 has moved, but is still PRINTed in the old position, you may shoot it dead, and it will reappear next go. To avoid this, we have this sequence: (1) PRINT both aliens; (2) You fire/move; (3) You fire/move; (4) Move both aliens, goto step 1. Now, whichever alien you fire at, it cannot be at one position and PRINTed at another.

200–320 PRINTs both aliens, 210 deals with the aliens' noise, 300 moves you, and makes a noise. 310–320 checks for firing, and has a firing sound. If you hit, goto 500, otherwise update 'shots left' at 550.

400 moves both aliens.

410 checks for alien landing and repeats sequence.

420–490 Alien lands, ambulance comes to drag you off, score and highscore given and new game offered.

500–540 You hit alien. Explosion and noise and bring new

alien on to top of screen at random position. Increment score, and missiles (if appropriate).

550–560 Update score and missiles left.

8000–8010 Set up user-defined graphics.

8500–8530 DATA for user-defined base, alien, missile and ambulance.

Listing

```
   1 REM   First Contact
   2 REM   @ S.Robert Speel 1982

  10 GO SUB 8000
  20 INK 0: PAPER 0: BORDER 0: C
LS
  30 LET hs=0
  40 LET sc=0: LET sh=20: LET t=
1
  50 PRINT INK 5;AT 20,10;CHR$ 1
45

 100 DIM x(6): FOR f=1 TO 4 STEP
 3: LET x(f)=INT (RND*25)+3: LET
 x(f+1)=1: LET x(f+2)=1: NEXT f:
 LET a=10

 200 FOR f=1 TO 4 STEP 3: PRINT
INK 4;AT x(f+1),x(f)-1;" ";CHR$
144;" ";AT x(f+1)-1,x(f)-x(f+2);
" ": NEXT f
 210 FOR f=1 TO 4 STEP 3: BEEP .
05,f

 300 LET rs=a: LET a=a+(INKEY$="
8")-(INKEY$="5")+(a<2)-(a>29): I
F rs<>a THEN BEEP .01,20: PRINT
INK 5;AT 20,a-1;" ";CHR$ 145;" "

 310 IF INKEY$="7" AND sh>0 THEN
 FOR g=18 TO 1 STEP -1: PRINT IN
K 2;AT g,a;CHR$ 146;AT g+1,a;" "
: BEEP .005,40: NEXT g: PRINT AT
 1,a;" ": LET sh=sh-1: LET t=1:
IF a=x(1) OR a=x(4) THEN LET sc=
sc+1: GO TO 500

 320 IF t=1 THEN GO TO 550
```

```
 400 NEXT f: FOR f=1 TO 4 STEP 3
: LET x(f+1)=x(f+1)+(RND<.5): LE
T x(f+2)=SGN (INT (RND*3)-1+(x(f
)<2)-(x(f)>28)): LET x(f)=x(f)+x
(f+2)
 410 NEXT f: IF x(2)<20 AND x(5)
<20 THEN GO TO 200

 420 PRINT PAPER 6;AT 10,1;"An a
lien has landed.";AT 12,4;"Your
score is ";sc: IF sc>hs THEN LET
 hs=sc
 430 PRINT INK 6;AT 14,5;"Best s
o far = ";hs

 440 FOR f=0 TO a-3 STEP 2:: PRI
NT INK 7;AT 20,f;"  ";CHR$ 147;C
HR$ 148: BEEP .6,18: BEEP .7,14.
5: NEXT f
 450 FOR f=1 TO 10: BEEP .5,-50+
f: NEXT f

 460 FOR f=a-3 TO 28 STEP 2: PRI
NT INK 7;AT 20,f;"  ";CHR$ 147;C
HR$ 148: BEEP .6,18: BEEP .7,14.
5: NEXT f
 470 PRINT INK 3+INT (RND*5);AT
18,0;"Another game? (Y/N)": IF I
NKEY$="y" THEN BEEP .5,10: CLS :
 GO TO 40

 480 IF INKEY$="n" THEN BEEP .5,
10: PAPER 6: BORDER 5: STOP
 490 BEEP .01,RND*50: GO TO 470

 500 FOR f=1 TO 10: PRINT INK RN
D*8;AT x(2)*(a=x(1))+(x(5) AND (
a=x(4))),a;"█";: BEEP .05,-RND*5
0: NEXT f: PRINT CHR$ 8;"  "
 510 IF a=x(4) THEN LET x(5)=1:
LET x(4)=INT (RND*20)+5
 520 IF a=x(1) THEN LET x(2)=1:
LET x(1)=INT (RND*20)+5
 530 IF a=x(4) THEN LET x(5)=1
 540 IF sc>5 THEN LET sh=sh+INT
(RND*3)

 550 PRINT INK 6;AT 0,0;"Score =
";sc;"   Shots left = ";sh;"  "
 560 LET t=0: GO TO 400

8000 FOR f=0 TO 4: FOR g=0 TO 7:
 READ a: POKE USR CHR$ (144+f)+g
,a: NEXT g: NEXT f
```

```
8010 RETURN

8500 DATA 129,169,219,126,60,24,
36,66
8510 DATA 0,0,36,36,36,60,126,12
6
8520 DATA 0,0,24,24,24,36,0,0
8530 DATA 31,63,126,124,126,127,
56,16,236,254,242,115,255,255,14
,4
```

User-defined Characters

All the letters and numbers, as well as the special graphic symbols and most of the other symbols, occupy one character square each. Each character square is made up of eight rows, each of eight dots, which can each be set to ink or paper. User-defined characters are characters where you, the user, design and program your own shapes. You have space for 21 different user-defined characters in any one program, though there are ways to get around this limitation and have as many as you want.

User-defined graphics are very useful in games – small green aliens, robots, spacecraft, games paddles and so on. For mathematical and scientific programs, Greek letters, square-root signs and large matrix brackets may be needed. When maps are made on screen, symbols can be designed to indicate marshland, rivers, built-up areas and so on. Graphs, pie-charts and filing systems may also use special symbols.

A user-defined character (user-defined graphic) is first designed as a black silhouette on an 8×8 grid of squares. Each row is then made into a binary number, a black square being a 1, and a white square, zero. So this 'graphic man' becomes 8 binary numbers, like this:
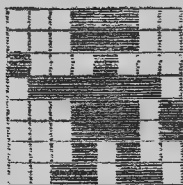

```
BIN 00011100
BIN 00011100
BIN 10001000
BIN 01111110
BIN 00011101
BIN 00011101
BIN 00010100
BIN 00110110
```

For shortness, these numbers can be converted to decimals –
it is not clear in the manual that this can be done. The code for
the graphic may be entered as a list of POKEs, like this:

```
 1 REM   User defined Man
10 POKE USR "A",28
20 POKE USR "A"+1,28
30 POKE USR "A"+2,136
40 POKE USR "A"+3,126
50 POKE USR "A"+4,29
60 POKE USR "A"+5,29
70 POKE USR "A"+6,20
80 POKE USR "A"+7,54
```

A simpler way is to use a FOR-NEXT loop, which READs the
numbers from a DATA statement elsewhere:

```
 1 REM   User defined Man

10 FOR f=0 TO 7: READ a: POKE
USR "A"+f,a: NEXT f
20 DATA 28,28,136,126,29,29,20
,54
```

For a number of different characters to be defined, an extra
loop can be added.

Here the 'A' is replaced by CHR$(144+f), which
incidentally avoids confusion between the capital A and graphic
A, which are indistinguishable on screen and in listings and
easy to forget. So for (e.g.) five user-defined graphics:

```
 1 REM   User defined Man,Woman
,Child,Cat and Dog

10 FOR f=0 TO 4: FOR g=0 TO 7:
READ a: POKE USR CHR$ (144+f)+g
,a: NEXT g: NEXT f
20 DATA 28,28,136,126,29,29,20
,54
30 DATA 8,29,137,126,28,62,20,
54
40 DATA 0,0,16,64,56,16,40,40
50 DATA 0,0,4,2,34,28,20,20
60 DATA 0,0,0,64,252,58,40,72
```
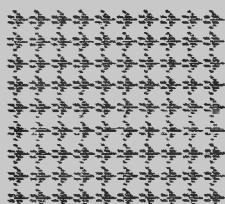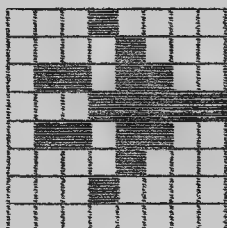
You can design your characters on squared paper, ruled into
8×8 square sections, but it is convenient to design the

character(s) on screen so that it is possible straight away to see what it looks like as a character in the actual size of the finished product, and be able to alter it until perfect.

```
Press keys 5,6,7,8 to move, key
0 for black square, 9 to erase
square, and 1 when finished.

   DATA 16,8,108,31,108,8,16,0

Corrections? (y/n)
```

User-defined Graphic Designer

To use
When you RUN this program, an 8×8 grid of squares appears at the top of the screen, with a marker in one corner.

To design a graphic, the marker is moved around the grid using keys 5, 6, 7 and 8. To fill in a square with black, press key 0. To wipe out a square which is filled in, press key 9.

When you have finished your graphic, press key 1. The DATA for the graphic will appear, together with a single actual-size version of your graphic and a block of them so that you can see what they look like in a group. You are then asked whether you wish to make any corrections, after seeing how your graphic really looks. Press key Y to put yourself back in command of the marker. If you wish to scrap the whole graphic, you will need to re-RUN the program.

Press keys 5,6,7,8 to move, key
0 for black square, 9 to erase
square, and 1 when finished.
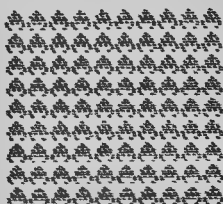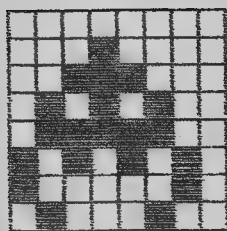
DATA 0,16,56,84,124,170,130,68

Corrections? (y/n)

Notes on listing
50 A$ will hold the binary for the graphic – initially all zeros.
60 Co-ords of marker on grid.
70 DRAW grid.
80 Print instructions.
100 DRAW marker on grid.
110 Wait for number key to be pressed.
130 Wipe out marker.
140 Move marker according to INKEY$.
150 If 0 pressed, goto 'ink in square' routine.
160 If 9 pressed, goto 'wipe square clean' routine.
170 Check for finished graphic.
190 Repeat loop.
200–280 Finished graphic. The DIMensioned array A holds
  the data, found by adding 'DIM' to A$ and finding its
  VALue.
  Note that DIM is a keyword, and must be entered as such,
  i.e., press CAPS SHIFT, then SYMBOL SHIFT, then let
  go and press the J key.
  Note use of CHR$ 8 in line 210 to backspace and remove the
  last comma in the DATA statement.
  If corrections, goto 300, otherwise STOP.

300 Clear bottom part of screen and go to main loop.
500–510 Black in square, and put a 1 in A$. Goto main loop.
600–610 Whiten square and put a 0 in A$. Goto main loop.

Listing

```
   1 REM U.D.G. Designer
   2 REM © S.Robert Speel 1982

  50 DIM a$(8,8): FOR f=1 TO 8:
LET a$(f)="00000000": NEXT f
  60 LET x=1: LET y=1
  70 FOR f=0 TO 8: PLOT 0,80+f*1
0: DRAW 80,0: PLOT f*10,80: DRAW
 0,80: NEXT f
  80 PRINT AT 13,0;"Press keys 5
,6,7,8 to move, key 0 for black
square, 9 to erase  square, and
1 when finished."

 100 OVER 1: PLOT (x-.6)*10,164-
y*10: DRAW 1,0: DRAW 0,2: DRAW -
2,0: DRAW 0,-2
 110 IF INKEY$<"0" THEN GO TO 11
0
 120 PLOT (x-.6)*10,164-y*10: DR
AW 1,0: DRAW 0,2: DRAW -2,0: DRA
W 0,-2: OVER 0
 130 LET x=x+(INKEY$="8")-(INKEY
$="5"): LET y=y+(INKEY$="6")-(IN
KEY$="7")
 140 LET x=x+(x<1)-(x>8): LET y=
y+(y<1)-(y>8)
 150 IF INKEY$="0" THEN GO TO 50
0
 160 IF INKEY$="9" THEN GO TO 60
0
 170 IF INKEY$="1" THEN GO TO 20
0
 180 IF INKEY$>" " THEN BEEP .5,
-50
 190 GO TO 100


 200 DIM a(8): PRINT AT 17,0;" D
ATA ";
 210 FOR f=1 TO 8: LET a(f)=VAL
("BIN "+a$(f)): PRINT a(f);",";:
 NEXT f: PRINT CHR$ 8;" "
 220 FOR f=0 TO 7: POKE USR CHR$
 144+f,a(f+1): NEXT f
```

```
 230 PRINT AT 2,30;CHR$ 144: PRI
NT AT 1,15;
 240 FOR f=1 TO 10: FOR g=1 TO 1
0: PRINT CHR$ 144;: NEXT g: PRIN
T AT f,15;: NEXT f
 250 PRINT AT 20,0;"Corrections?
 (y/n)"
 260 LET b$=INKEY$: IF b$="y" TH
EN GO TO 300
 270 IF b$="n" THEN STOP
 280 GO TO 260

 300 FOR f=17 TO 20: PRINT AT f,
0;"
   ": NEXT f: GO TO 100

 500 FOR f=1 TO 9: PLOT (x-1)*10
+1,170-y*10-f: DRAW 8,0: NEXT f
 510 LET a$(y,x)="1": BEEP .5,-2
0: GO TO 100

 600 INVERSE 1: FOR f=1 TO 9: PL
OT (x-1)*10+1,170-y*10-f: DRAW 8
,0: NEXT f: INVERSE 0
 610 LET a$(y,x)="0": BEEP .5,-2
0: GO TO 100
```

Graphic Lists

Naturally, you will want to make your own characters or graphics, but here are some useful ones to start you off. The eight numbers for each graphic are given as a DATA list.

½ ½ ½ ½ ½ ½
 DATA 64,64,68,72,22,34,4,6

⅓ ⅓ ⅓ ⅓ ⅓ ⅓
 DATA 64,68,72,86,33,70,1,6

¼ ¼ ¼ ¼ ¼ ¼
 DATA 64,68,72,80,40,74,15,2

⅓ ⅓ ⅓ ⅓ ⅓ ⅓
 DATA 192,68,136,214,33,70,1,6

¾ ¾ ¾ ¾ ¾ ¾
 DATA 192,32,195,40,212,37,71,1

↑ ↑ ↑ ↑ ↑ ↑
```
DATA 0,16,56,84,16,16,16,0
```

↓ ↓ ↓ ↓ ↓ ↓
```
DATA 0,8,8,8,42,28,8,0
```

→ → → → → →
```
DATA 0,0,8,4,126,4,8,0
```

← ← ← ← ← ←
```
DATA 0,16,32,126,32,16,0,0
```

↗ ↗ ↗ ↗ ↗ ↗
```
DATA 0,30,6,10,18,32,0,0
```

↘ ↘ ↘ ↘ ↘ ↘
```
DATA 0,0,32,18,10,6,30,0
```

↙ ↙ ↙ ↙ ↙ ↙
```
DATA 0,0,4,72,80,96,120,0
```

↖ ↖ ↖ ↖ ↖ ↖
```
DATA 0,120,96,80,72,4,0,0
```

α α α α α α
```
DATA 0,0,4,52,72,72,54,0
```

β β β β β β
```
DATA 24,36,56,36,36,56,32,32
```

γ γ γ γ γ γ
```
DATA 2,34,18,20,8,20,24,0
```

δ δ δ δ δ δ
```
DATA 8,20,16,8,24,36,24,0
```

θ θ θ θ θ θ
```
DATA 24,36,68,102,92,68,56,0
```

∅ ∅ ∅ ∅ ∅ ∅
```
DATA 1,26,36,74,82,36,88,128
```

μ μ μ μ μ μ
```
DATA 0,0,0,40,40,56,36,64
```

π π π π π π
```
DATA 0,0,2,124,168,40,40,0
```

ƒ ƒ ƒ ƒ ƒ ƒ
```
DATA 12,18,80,56,20,16,144,96
```

⇄ ⇄ ⇄ ⇄ ⇄ ⇄
```
DATA 0,8,4,126,1,126,4,8
```

```
DATA 0,0,0,24,102,0,126,0

DATA 0,1,2,2,36,20,8,0

DATA 0,62,42,62,20,28,62,42

DATA 0,24,126,255,24,36,0,0

DATA 24,60,90,126,24,36,66,0

DATA 0,56,84,56,124,170,170,170

DATA 24,36,24,90,189,153,165,12
9
```

Customized Graphic Sets

Some types of graphics tend to be used together; for example, a spaceship graphic is likely to be accompanied by several other spacecraft graphics, missiles and 'explosion' graphics.

A couple of sets are given here. It may be more convenient to store these sets on a separate cassette of their own, so that they can be easily located and MERGEd with other programs. For this reason, the line numbers start at 9300, since it is unlikely that your main program will have the line numbers 9300 onwards already in use.

```
9300 REM Tank UDGs by S.R.S.

9310 RESTORE 9320: FOR f=0 TO 15
: FOR g=0 TO 7. READ a: POKE USR
 CHR$ (144+f)+g,a: NEXT g: NEXT
f

9320 DATA 0,0,0,30,56,126,255,10
2,0,0,0,120,28,126,255,102
9330 DATA 0,0,0,0,120,70,127,107
,0,0,0,0,30,98,254,102
```

```
9340 DATA 0,0,63,1,15,127,42,31,
0,128,192,240,248,255,170,252
9350 DATA 0,1,3,15,31,255,85,63,
0,0,252,128,240,254,84,248
9360 DATA 0,0,14,18,63,63,21,8,0
,0,0,2,252,252,84,248
9370 DATA 0,0,0,64,63,63,42,31,0
,0,112,72,252,252,168,16
9380 DATA 0,31,2,7,15,127,42,31,
128,0,128,224,240,252,168,240
9390 DATA 1,0,1,3,15,63,21,15,0,
248,84,224,240,254,84,248

9400 PRINT "Tank UDG Set"''
9410 FOR f=144 TO 159: PRINT CHR
$ f;" " AND f/2<>INT (f/2);" " A
ND f<148;: NEXT f: PRINT ''
9420 FOR f=65 TO 80: PRINT CHR$
f;" " AND f/2=INT (f/2);" " AND
f<69;: NEXT f
```

```
Tank UDG Set
```

A B   C D   EF GH IJ KL MN OP

```
9300 REM   Castle UDGs by SRS
9310 FOR f=0 TO 14: FOR g=0 TO 7
: READ a: POKE USR CHR$ (144+f)+
g,a: NEXT g: NEXT f

9320 DATA 204,204,255,85,255,255
,255,255,126,102,255,85,255,255,
255,255
9330 DATA 108,124,187,255,187,25
5,187,255
9340 DATA 0,51,51,63,22,22,30,31
,0,0,0,0,0,102,102,255,0,204,204
,252,104,104,120,248

9350 DATA 219,219,255,223,255,22
3,255,223,153,255,255,231,195,19
5,195,195,216,216,255,251,255,25
1,255,251
9360 DATA 0,219,219,255,231,126,
102,126,16,16,56,56,108,108,254,
84
```

```
9370 DATA 195,219,255,102,126,12
6,86,107,195,219,255,102,126,126
,106,214
9380 DATA 127,103,255,85,255,255
,255,255,254,230,255,85,255,255,
255,255

9400 PRINT "Castle UDGs"'': FOR
f=144 TO 158: PRINT CHR$ f;" ";:
 NEXT f: PRINT
9410 FOR f=144 TO 158: PRINT CHR
$ (f-79);" ";: NEXT f
```

Castle UDGs
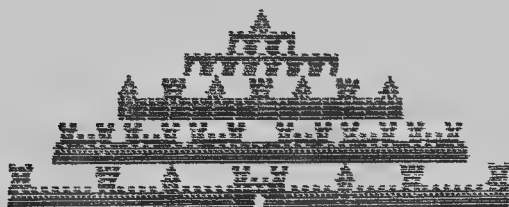


```
   10 REM   Castle picture by SRS

   20 PRINT TAB 14;"K";TAB 13;"HH
H";TAB 11;"HHHHHHH"
   30 PRINT TAB 8;"K J K J K J K"
   40 PRINT TAB 8;"GGGGGGGGGGGGGG"
   50 PRINT TAB 5;"DEFDEFDEF DEFD
EFDEF"
   60 PRINT TAB 5;"NONONONONNNONON
ONONO"
   70 PRINT TAB 3;"L    J   K   DEF
  K  J    H"
   80 PRINT TAB 3;"OAAABAACAAGHIA
ACAABAAAN"
```

Graphics Rotation and Reflection

This is a very useful program when you have designed some
user-defined graphics and now want them facing in different
directions. A spaceship may be needed pointing into all four
points of the compass; a man may have to fall off a cliff
horizontally or in an upside-down position; and a dinosaur
facing right has to have an opponent facing left. This program
gives you graphic rotation in all four directions and also
reflected in all four directions – eight poses in all.

To use
RUN and input the data for your graphic one by one, with
ENTER after each. You can input it as normal numbers (base
10), e.g., 40, or as a binary number, e.g., BIN 101000. When
you have finished, the screen clears, and after a short pause a
large version of your graphic appears, along with an actual-size
character showing your graphic singularly and in a block of
nine copies of it. The data is also given.

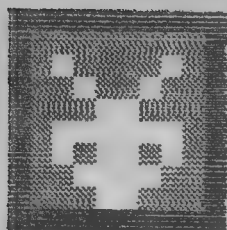Press ENTER and the reflected upside-down version of your
graphic appears, with data, etc. When you are ready, press
ENTER, and repeat for all the other reflections and rotations.

Typical use of program
Due to symmetry of this graphic, there are only four ways of
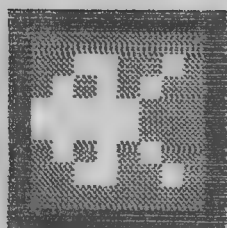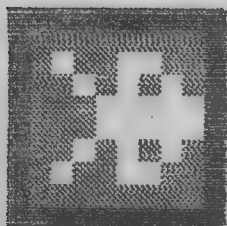pointing it.
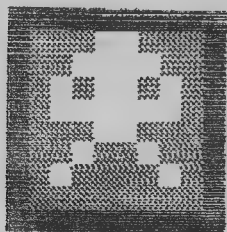
Data 24,60,90,126,24,36,66,0

Data 0,66,36,24,126,90,60,24

Data 0,50,84,248,248,84,50,0

Data 0,76,42,31,31,42,76,0

Notes on listing

50 A$ holds the picture of the graphic.

60 CHR$ 144 is the 'chessboard' pattern which appears grey on the picture.

70 B$ holds the binary numbers which represent the graphic.

100–160 INPUT data for your graphic. Set A$ to picture of graphic.

200–260 Draw picture of graphics and actual graphic characters.

210 READ columns and rows of A$ to be printed. This READing of variables, rather than numbers or strings, can be very useful. However, as each pair has to be read several times, RESTORE is used and hence each pair must have its own line number.

220 Set B$ to binary number representation of A$.

240 The BIN is the keyword BIN, *not* 3 separate letters.

250 PRINT block of characters and single character.

260 Repeat for next rotation/reflection when key pressed.

8000–8070 DATA for column and row of A$ to print.

Note Do not put all the DATA into one program line.

Listing

```
    1 REM Graphics Rotation and
            Reflection
    2 REM  © S.Robert Speel 1982

  10 INK 1: PAPER 6: BORDER 4: C
LS

  50 DIM a$(8,8): FOR f=1 TO 8:
FOR g=1 TO 8: LET a$(f,g)=CHR$ 1
44: NEXT g: NEXT f
  60 FOR f=0 TO 7 STEP 2: POKE U
SR CHR$ 144+f,85: POKE USR CHR$
144+f+1,170: NEXT f
  70 DIM b$(8,8)

 100 PRINT " Input the data for
your"'"graphic, following each n
umber"'"with ENTER."
 110 FOR f=1 TO 8: INPUT "Number
";(f);" ";nu: BEEP .3,10
```

```
 120 FOR g=0 TO 7: IF nu>=2↑(7-g
) THEN GO TO 140
 130 GO TO 150
 140 LET a$(f,g+1)=" ": LET nu=n
u-2↑(7-g)
 150 NEXT g: NEXT f
 160 CLS

 200 FOR f=0 TO 7: PRINT "██████
███"
 210 FOR g=1 TO 8: PRINT "█";: F
OR h=1 TO 8: RESTORE 8000+f*10:
READ a,b: PRINT a$(a,b);
 220 LET b$(g,h)="0": IF a$(a,b)
=" " THEN LET b$(g,h)="1"
 230 NEXT h: PRINT "█": NEXT g:
PRINT "█████████"
 240 PRINT AT 12,0;"Data ";: FOR
 i=1 TO 8: PRINT VAL ("BIN "+b$(
i));",";: POKE USR CHR$ 145+i,VA
L ("BIN "+b$(i)): NEXT i: PRINT
CHR$ 8;" "
 250 PRINT AT 5,15;CHR$ 145: FOR
 i=7 TO 9: PRINT AT i,20;CHR$ 14
5;CHR$ 145;CHR$ 145: NEXT i
 260 PAUSE 0: CLS : NEXT f

8000 DATA g,h
8010 DATA 9-g,h
8020 DATA g,9-h
8030 DATA 9-g,9-h
8040 DATA h,g
8050 DATA 9-h,g
8060 DATA h,9-g
8070 DATA 9-h,9-g
```

Extending User-defined Graphics

There is a limit to the number of user-defined graphics which can be used at one time – 21. You may think it unlikely that you will want more than this number in one program, but there are many situations where you will find you will need a few extra graphics. For instance, say in a 'space' game, your ship consists of two graphics: if you wish to rotate it in all four points of the compass, you will need a total of eight user-defined graphics. Add a couple of enemy spacecraft and you may not have any space left for user-defined missiles . . . .

Another case is when you want a complete user-defined alphabet, for example Greek. One way of coping, if all the graphics are not to be used at once, is to use a 'Data switch'.

Data switch means that where there is an alternative between several different graphics, according to the situation, the program RESTOREs to the line number of the graphic(s) desired. So the same user-defined character space, e.g., CHR$ 144, can be used for several different graphics. This is a useful technique to cut down on sub-routines even if you do not intend to use many user-defined graphics. A good example of this method is found in Space Raider.

Another problem with user-defined graphics is recognizing them. SCREEN$ will not work, but just return the empty string. The normal way to get around this is to use ATTR, and have each of the various user-defined graphics characters a different colour or brightness to distinguish them. However, this will not cover all eventualities. Consider the common problem of background. A user-defined tank is travelling across a terrain of full stops and plus signs. To stop the tank leaving an empty stripe behind it as it moves, the square in front of it is deleted each go and copied into A$, so that, for example,

a tank going across the screen to the right, PRINTed at Y,X, a suitable command would be:

LET A$ = SCREEN$(Y,X+1)

Then, when the tank moves on, A$ is PRINTed in its place, so the tank moves 'over' the terrain. But when the full stops and plus signs are replaced by user-defined grass and bushes, the above technique will not work as SCREEN$ cannot detect the user-defined graphic. Again, we can use ATTR: if the ATTRibutes of the square add up to 116 (bright green on yellow) PRINT a bush, if 52 (dull green on yellow) PRINT grass. This becomes impracticable when there are several different types of foliage or other graphics to be recognized.

What we need is a way to identify such UDGs using SCREEN$. There is a system variable called CHARS which stores the location of the beginning of the character set in ROM −256. If CHARS is POKEd to a new value, in RAM, we can make our own character set. Usually, we will want to keep the normal character set, but change a few of the less-often used characters, such as the ampersand (&), the exclamation mark (!) and the £ sign. These new graphics will be recognized by SCREEN$ and thus be more useful than normal user-defined graphics. In fact, if you want to keep your £ sign, exclamation mark and so on, you can set up some normal user-defined graphics to hold these.

The idea of using an alternate character set in RAM is also a solution to the problem of not having enough user-defined graphics. You can replace not only the little used characters, but also, for instance, the lower-case letters. You can have your complete upper- and lower-case Greek alphabet and keep all 21 of your normal user-definable graphics available!

The only problem is with listings. After having changed your complete character set to Greek, all the error messages and listings will be in Greek letters. The error messages do not really matter, but a listing in Greek (or in block graphics, etc.) can be difficult to follow and correct. There is a cure.

At the end of the program, have a line POKEing CHARS

back to its normal position in ROM.

The following program, Letterwriter, shows how you can change your character set to 'handwritten' text.

Letterwriter

To use

Letterwriter is a program which lets you write short letters using simulated handwriting. All the normal letter characters, both capital and lower case, are replaced by letters designed to join up to each other.

When you RUN this program the first time after typing it in or LOADing it, the name 'Letterwriter' will appear on screen, and under it a constantly increasing number. This number will rapidly increase to 760, and it is there so that you can tell that the program is running. Then, after a short pause, you will be asked how you sign yourself, who the letter is to be addressed to, and the date (if no date is required just press ENTER).

Next you are asked for the text of the letter. Type it in without worrying about overlapping from one line to the next in mid-word – the computer will take care of that. When you have finished, press ENTER and, after a few seconds, the whole letter will appear, in 'handwriting', including a flourish on your signature.

Dear Reader,

Do you feel that your ZX Spectrum prints too soullessly? Are the lines of text too inhuman for you? If so, use

LETTER WRITER to get warm user-friendly "handwriting".

Yours,

S.Robert Speel

1 April 1996

Dear Sir,

We regret to inform you that your computer is once more delayed, due to rust in the chips. But you will get it soon, honestly.

Yours,

J Smith (Consumer Dept)

Notes on listing

The main part of this program is the data in lines 9400–9630. These contain the data for the alternate letters, and may be used in other programs. The rest of the program consists of routines to POKE the data into RAM, the part where you enter your letter and what is really a very simple 'word processor' to set out the letter neatly, without words overlapping lines.

20 This line means that after the first RUN, you will not have to wait while the character set is POKEd into RAM.

110–150 Input name, who the letter is to, date and text of letter.

Note that the text, contained in d$, has some dummy spaces put on to the end of it, as d$ tends to grow in length as it is processed.

160 POKE CHARS to the new character set in RAM.

Note that CHARS can be POKEd from character set to character set without delay at any time.

200–320 Check end of each line of d$ (text of letter) and if a word is split there, add spaces until the word is all on the next line.

Note that this results in d$ increasing in length. Finally, the excess spaces at the end are removed.

400–420 PRINT text of letter, together with date, signature, etc.

Note that the signature is complete with a curly flourish at the end!

As listed, the letter will leave a blank line between each line of text, but for longer letters, you can make the computer print on every line by erasing the two apostrophes before the 'NEXT F' in line 410.

430 POKE CHARS back to Sinclair's character set.

440–490 COPY routine – delete if you have no printer.

9300–9320 POKE complete character set into RAM starting at address 31000. Print out every tenth byte to let you know how it is getting on.

9330 POKE your lower-case letters into RAM, and the three-character flourish for signature.

9340 POKE your capital letters into RAM. RESTORE not necessary but useful as a safeguard if more DATA added to program later.

9350 Go back to main program.

9360 DATA for flourish.

9400–9520 DATA for lower-case letters.

9550–9630 DATA for capital letters.

Listing

```
   1 REM Letter Writer
   2 REM  © S.Robert Speel 1982

  10 PRINT "Letter Writer"
  20 IF PEEK 31500=0 THEN GO SUB
9300

 100 CLS : PRINT "Letter Writer"
 110 INPUT "How will you sign yo
urself? ";a$
 120 INPUT "Who is the letter to
?";b$
 130 INPUT "Today's date?";c$
 140 INPUT "type in text of lett
er ";d$: LET d$="       "+d$
 150 FOR f=1 TO 5: LET d$=d$+"
        ":
 NEXT f
 160 POKE 23606,24: POKE 23607,1
20

 200 FOR f=1 TO LEN d$-32 STEP 3
2
 210 IF d$(f+31)=" " THEN GO TO
310
 220 FOR g=f+31 TO f STEP -1
 230 IF d$(g)=" " THEN GO TO 300
 240 NEXT g
 250 GO TO 310

 300 FOR h=g TO f+31: LET d$=d$(
 TO g)+" "+d$(g+1 TO ): NEXT h
 310 NEXT f
 320 LET d$=d$( TO LEN d$-128)

 400 CLS : PRINT ,c$'TAB 4;"Dear
 ";b$;", "
 410 PRINT '': FOR f=1 TO LEN d$
-32 STEP 32: PRINT d$(f TO f+31)
'': NEXT f
 420 PRINT ''TAB 10;"Yours,"''',a
$;"↑": PRINT TAB 14+LEN a$;"£_"
 430 POKE 23606,0: POKE 23607,60
 440 PRINT AT 21,0;"Copy? (y/n)"
 450 IF INKEY$="y" THEN GO TO 46
0
 460 IF INKEY$="n" THEN STOP
 470 GO TO 440
 480 PRINT AT 21,0;"            "
 490 COPY : STOP
```

```
9300 FOR f=0 TO 768: POKE 31000+
f,PEEK (15616+f)
9310 IF f/20=INT (f/20) THEN PRI
NT AT 1,0; f
9320 NEXT f
9330 RESTORE 9350: FOR f=496 TO
728: READ a: POKE 31000+f,a: NEX
T f
9340 RESTORE 9550: FOR f=264 TO
464: READ a: POKE 31000+f,a: NEX
T f
9350 RETURN

9360 DATA 0,0,48,72,136,16,96,12
8,1,134,120,0,0,224,16,32,0,7,24
,32,64,64,35,28

9400 DATA 0,0,56,4,62,197,60,0,1
6,48,32,124,163,34,60,0
9410 DATA 0,0,28,32,96,163,28,0,
4,4,4,62,69,196,60,0
9420 DATA 0,0,56,68,249,66,60,0,
12,10,20,112,159,48,60,96
9430 DATA 0,0,28,36,93,134,4,24,
0,48,80,95,120,203,76,0
9440 DATA 0,16,0,16,49,214,8,0,1
6,0,16,48,211,60,80,96

9450 DATA 32,80,116,164,56,37,38
,0,24,40,40,48,80,145,14,0
9460 DATA 0,0,104,84,212,85,86,0
,0,0,0,56,100,165,38,0
9470 DATA 0,0,56,70,197,68,56,0,
0,0,56,100,167,60,32,32
9480 DATA 0,0,56,100,188,5,6,4,0
,0,36,122,161,32,32,0
9490 DATA 0,0,48,64,187,4,56,0,3
2,32,56,32,97,166,24,0
9500 DATA 0,0,36,100,164,37,26,0
,0,0,36,38,105,168,16,0
9510 DATA 0,0,34,106,171,42,20,0
,0,0,102,152,136,25,102,0
9520 DATA 0,0,36,100,191,20,36,2
4,0,0,56,72,145,34,116,12

9550 DATA 24,36,36,60,36,165,102
,0,120,36,36,56,37,166,120,0,24,
36,36,32,96,167,24,0
9560 DATA 60,82,81,57,97,209,62,
0,56,72,64,48,64,199,56,0,31,16,
48,95,144,16,16,0
9570 DATA 56,68,68,64,207,68,56,
0,100,36,36,60,36,165,102,0,58,8
4,16,16,16,58,84,0
```

```
9580 DATA 63,22,8,8,72,72,48,0,9
8,164,40,48,40,165,98,0,96,160,9
6,32,32,33,126,0
9590 DATA 34,54,42,42,34,163,98,
0,108,178,162,34,34,34,35,0,24,3
6,36,36,101,166,56,0

9600 DATA 60,34,34,124,160,32,32
,0,56,68,68,68,212,76,60,2,56,36
,36,56,100,165,38,0
9610 DATA 28,34,32,92,130,3,28,0
,113,142,8,8,9,14,24,0,68,68,68,
68,68,197,62,0
9620 DATA 68,68,70,69,168,40,16,
0,130,130,130,147,146,146,108,0,
130,68,40,16,16,170,68,0
9630 DATA 18,18,50,94,130,15,18,
12,62,68,8,16,32,77,242,0
```

High-resolution Graphics

By high-resolution graphics I mean the Spectrum's ability to draw fine lines, curves and circles, and PLOT on a grid of 192×256 dot positions, not user-defined graphics, which are dealt with on pp. 34–56.

When you want fast moving graphics, use user-defined graphics. In many programs the moving bits will be user-defined graphics and the backgrounds may be drawn with high-resolution graphics.

High-resolution is suitable for picture drawing, graph plotting, plans of buildings, backgrounds, maps and pie-charts. It is also useful for illustrating many maths and scientific programs.

Geometrical shapes and patterns can be most useful in 'showing off' the Spectrum's powerful high-res. Here are a few short programs which are worth experimenting with.

Cage

There is a great sense of depth to this picture on screen. It is easy to alter the size and shape by changing a and b. By changing the STEP value, line 30, you can achieve a more solid shape (STEP 2 is quite effective) or a more 'wiry' one (try STEP 4).

This is a good program to experiment with, and to see how to alter the depth, shape and structure of a picture on screen.
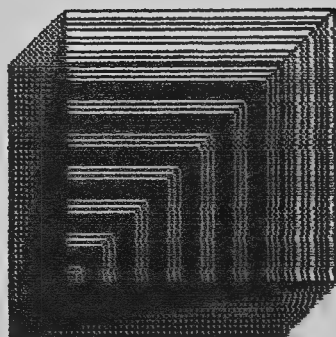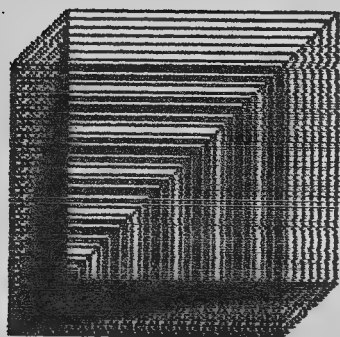
Listing

```
   1 REM   Cage

  10 BRIGHT 1: PAPER 6: INK 1: B
ORDER 2: CLS
  20 LET x=60: LET y=25
  30 FOR f=1 TO 100 STEP 2.5
  40 LET a=100-f: LET b=100-f: L
ET c=(a+b)/10

 100 PLOT x,y
 110 DRAW a,0: DRAW 0,b: DRAW -a
,0: DRAW 0,-b
 120 DRAW c,c: DRAW 0,b: DRAW -c
,-c: DRAW 0,-b

 200 PLOT x+c,y+c
 210 DRAW a,0: DRAW 0,b: DRAW -a
,0: DRAW 0,-b

 300 PLOT x+a,y
 310 DRAW c,c: DRAW 0,b: DRAW -c
,-c: DRAW 0,-b
 320 NEXT f
```
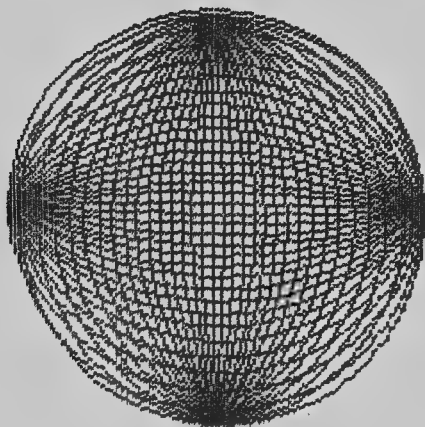
Sphere

This program shows use of high-res DRAWing.

The picture on paper does not, unfortunately, compare with the screen picture, which looks very solid and almost three-dimensional.

Notes on listing

Lines 120 and 170 show DRAW forming curves. If these are changed to DRAW 100,0,f and DRAW 0,–100,f, and the second PLOT (line 160) changed, a smaller sphere can be drawn. Try altering the STEP values, in lines 100 and 150, to make the sphere 'darker' and more solid, or 'lighter' and more segmented.

Listing

```
   1 REM sphere
  10 INK 3: PAPER 0: BORDER 2: C
LS
 100 FOR f=-PI TO PI STEP .2
 110 PLOT 50,65
```

```
120 DRAW 150,0,f
130 NEXT f

150 FOR f=-PI TO PI STEP .2
160 PLOT 120,160
170 DRAW 0,-150,f
180 NEXT f
```

Revolving Surface

With small shapes, DRAWing is fast enough to give a fairly
continuous moving picture.

Here a small flat surface appears to rotate 'into' the screen.

Notes on listing

Line 20 a controls the width difference of the rotating surface
in comparison to its height, b gives the speed and continuity
of the rotation, and c is the size of the shape.

Lines 100–130 contain a loop which DRAWs OVER 1 twice,
thus first drawing the surface, then erasing it. This gives a slight
flicker to the picture.

If c is increased, the 'surface' gets larger, and the flicker more
pronounced; if smaller, there is less flicker.
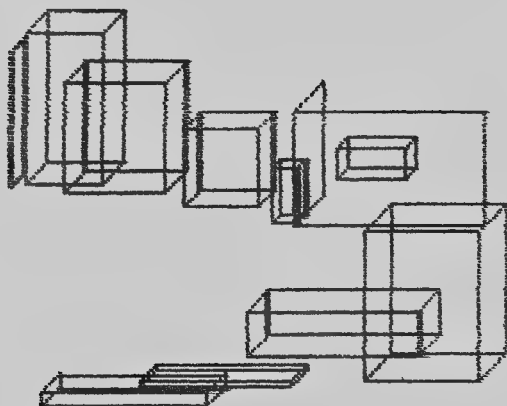
Listing

```
   1 REM Revolving Surface

  10 PAPER 6: BORDER 4: INK 0: C
LS : OVER 1
  20 LET a=0: LET b=10: LET c=20

 100 FOR f=1 TO 2
 110 PLOT 120+.5*a,90
 120 DRAW c-a,0: DRAW 0,c: DRAW
a-c,0: DRAW 0,-c
 130 NEXT f

 200 LET a=a+b
 210 IF a=c OR a<0 THEN LET b=-b
 220 GO TO 100
```

Block Complex

This program is based on a cube-drawing routine – by adding random length, breadth and position, the program draws a wide range of different blocks on the screen. Although 100 blocks can be drawn, the picture often looks at its best after about 20 or 30 blocks.

Notes on listing

20 Number of blocks, change as desired, or use BREAK.

30 x and y are the position of the bottom front left-hand corner of the block.

40 a is the length, b the breadth and c the depth of the block. Try making c independent of a and b (e.g., LET C = RND *15).

100–130 Draw block. Try making the ink colour random, or the paper colour, and see what happens as the screen fills up.

Listing

```
   1 REM Block Complex

  10 BRIGHT 1: PAPER 6: INK 1: B
ORDER 2
  20 FOR f=1 TO 100
```
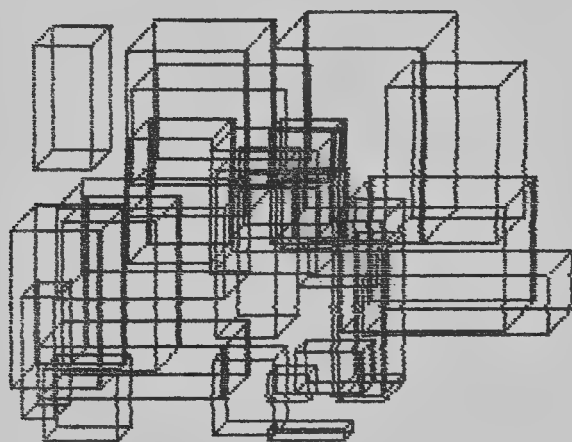
```
  30 LET x=RND*150: LET y=RND*90
  40 LET a=RND*70: LET b=RND*60:
LET c=(a+b)/10

 100 PLOT x,y: DRAW a,0: DRAW 0,
b: DRAW -a,0: DRAW 0,-b
 110 DRAW c,c: DRAW 0,b: DRAW -c
,-c: DRAW 0,-b
 120 PLOT x+c,y+c: DRAW a,0: DRA
W 0,b: DRAW -a,0: DRAW 0,-b
 130 PLOT x+a,y: DRAW c,c: DRAW
0,b: DRAW -c,-c: DRAW 0,-b
 140 NEXT f
```

Modules: Movement of a Small Object

In most moving graphics games the player is in control of some small object – representing a spacecraft, a car, a bat, etc., which can move at least in one dimension and often in two. By a small object I mean something up to about three graphics characters in size. For the following routines, an asterisk (*) denotes the object.

Movement in one dimension

This involves moving left and right, or up and down, not both.

Before actually writing a routine, we must consider what happens when the object moves. Assuming that we don't want to leave a trail of asterisks behind whenever the object moves, we have to erase the 'old' asterisk when moving to a different co-ordinate. Using CLS is a poor answer, since usually there are other things on the screen and to rePRINT or rePLOT these each go is unacceptable in terms of time taken. For left and right movement, if instead of just PRINTing *, we put a space before and after the asterisk, then when moving, the old asterisks are erased by the spaces. This is very suitable for games using 'paddles' or bats in breakout-type games, tennis games and so on, and also in games involving laser guns which move from side to side.

(A) LEFT/RIGHT MOVEMENT AT ONE SPEED MODULE

Note that the asterisk cannot go offscreen because of the $(X<2)$ and $(X>29)$. If instead of an asterisk you have, say, a paddle

```
MAINLOOP
     .
     .
 500 LET X=X+(INKEY$="8")-(INKEY
$="5")+(X<2)-(X>29)
 510 PRINT AT Y,X;" * "
     .
     .

GOTO MAINLOOP
```

made of three inverse spaces, then line 510 would read PRINT
AT Y,X;' ■■■ ' and the limits for X would be changed to
(X<3) and (X>28).

(B) UP/DOWN MOVEMENT AT ONE SPEED MODULE

```
MAINLOOP
     .
     .
 500 LET Y=Y+(INKEY$="6")-(INKEY
$="7")+(Y<2)-(Y>19)
 510 PRINT AT Y-1,X;" ";AT Y,X;"
*";AT Y+1,X;" "
     .
     .

GOTO MAINLOOP
```

Remember that the Y co-ordinate cannot be more than 21. To
PRINT a vertical line of, for instance, three asterisks, line 510
would become:

   510 PRINT AT Y-2,X;" ";AT Y-1,X;"*";AT Y,X;"*
   ";AT Y+1,X;"*";AT Y+2,X;" "

and the limits in line 500 changed to (Y<3) and (Y>18).
   Another common situation occurs when the object is moving
at a definite speed in one dimension and is controllable in
another dimension. For example, an aeroplane might go at a
constant speed across the screen, but its height is controlled by
the player.

(C) MOVEMENT IN TWO DIMENSIONS WITH CONTROL IN
ONE DIMENSION

```
MAINLOOP
       .
       .
       .
 500 LET Y=Y+(INKEY$="6")-(INKEY
$="7")
 510 LET X=X+1
 520 PRINT AT Y-1,X;"   ";AT Y,X;
" *";AT Y+1,X;"  "
 530 IF Y<2 OR Y>18 THEN GOTO CR
ASHROUTINE
 540 IF X>28 THEN PRINT AT Y,X;"
 "
 550 IF X>28 THEN LET X=0
       .
       .
       .
 GOTO MAINLOOP
```

(D) TWO-DIMENSIONAL MOVEMENT AT ONE SPEED

```
MAINLOOP
       .
       .
       .
 500 LET X=X+(INKEY$="8")-(INKEY
$="5")+(X<2)-(X>29)
 510 LET Y=Y+(INKEY$="6")-(INKEY
$="7")+(Y<2)-(Y>19)
 520 PRINT AT Y-1,X;"   ";AT Y,X
;" * ";AT Y+1,X;"    "
       .
       .
       .
 GOTO MAINLOOP
```

Here the 5, 6, 7 and 8 keys are all used to control the object.
The idea of erasing old copies of the object by PRINTing spaces
is more complex, since we need to print spaces all around the
object.

(E) MOVEMENT AT ONE SPEED WITH MOMENTUM

```
MAINLOOP
     .
     .
 500 LET X1=SGN (X1+(INKEY$="8")
-(INKEY$="5")+(X<2)-(X>29))
 510 LET Y1=SGN (Y1+(INKEY$="6")
-(INKEY$="7")+(Y<2)-(Y>19))
 520 LET X=X+X1
 530 LET Y=Y+Y1
 540 PRINT AT Y-1,X;"    ";AT Y,X
;"  *  ";AT Y+1,X;"
     .
     .
 GOTO MAINLOOP
```

When a spaceship moves in a particular direction, it will
continue to move in this direction until stopped. This is the
basis for momentum movement. If you make your object move
left by pressing key 5 it will carry on going in that direction until
you press 8 to stop it. If you press 5 then 7, the ship will go left
and up, diagonally.

Here we have to introduce two new variables: the velocity in
X, X1 and the velocity in Y, Y1. Thus, each go, the object will
move from (Y,X) to (Y+Y1, X+X1). Set X1 and Y1 to 0
initially.

(F) MOVEMENT WITH MOMENTUM AT MORE THAN ONE
SPEED

```
MAINLOOP
      .
      .
 500 LET X1=X1+(X1<-2)-(X1>2)+(I
NKEY$="8")-(INKEY$="5")+(X<2)-(X
>29)
 510 LET Y1=Y1+(Y1<-2)-(Y1>2)+(I
NKEY$="6")-(INKEY$="7")+(Y<2)-(Y
>19)
 520 LET X=X+X1
 530 LET Y=Y+Y1
 540 IF X<0 OR X>31 OR Y<0 OR Y>
21 THEN GOTO CRASHROUTINE
```

```
550 PRINT AT Y-Y1,X-X1;" ";AT Y
,X;"*"
       .
       .
       .
GOTO MAINLOOP
```

This means that if you hold down the 5 button the object will not only move, but accelerate.

The maximum speed allowed is 3 in any direction, but this can be changed to any desired value.

As for limits, if the object attempts to go offscreen at speed 1, it will merely bounce back, but if it goes offscreen at speed 2 or 3, it will crash – in a prepared CRASHROUTINE.

Notice here that, instead of PRINTing spaces round the object – there are now too many to do – a space is PRINTed on top of the object, erasing it, and then the object is PRINTed at its new position.

Of course, the boundaries of the screen in which the object is allowed to move in and the results of trying to move offscreen will be varied from game to game.

The above modules can be combined with other factors, such as SCROLL. The best way to do this is to put the routine in unaltered and see if something odd happens – typically a trail of 'bits of object' appearing – and this will tell you where to put more spaces in.

Avoiding Problems with Programs

As soon as you start altering programs or writing your own, you will find that your programs need debugging. In a 2K program there might be a dozen errors and, as you correct each one, the program will crash at a new point. This can be very disheartening, but there is a solution: do not write in a program more than ten lines long! (This applies only to your new programs or modification – existing written programs of any length can be copied.) Although this may sound odd, it is very useful. Instead of converting your 8K Monopoly-type game into a 16K version in one go, do it a little at a time. You might start by adding 'Community Chest' cards – there were only 'Chance' cards before – by delegating squares 6, 14 and 35 as 'Community Chest' squares. Each time a piece moves, if it lands on a 'COM CHEST' then a sub-routine at 8400 is consulted. For example:

```
4370 IF A(N) = (6 OR 14 OR 35) THEN GOSUB 8400
8400 PRINT "YOU LAND ON COMMUNITY CHEST"
8407 STOP
8500 RETURN
(A(N) is the position of the piece just moved.)
```

Now RUN the program with this addition to check that it works. Since there is little chance of landing on a 'Community Chest' each go, STOP the program, type LET A(N) = 6, NEWLINE, then GOTO 4370. When you find that 'YOU LAND ON COMMUNITY CHEST' is not printed, you can stop the program and find out why, with the considerable advantage of knowing that the error can only lie in one of lines 4370, 8400, 8407 and 8500. In this case the error is obviously

in line 4370. After experimentation, it is found that IF A(N) = (6 OR 14 OR 35) means IF A(N) = 1, you might decide to change line 4370 to IF A(N) = 6 OR A(N) = 14 OR A(N) = 35 THEN GOSUB 8400. Once you find that this works, carry on with a few more lines.

Obviously, just part of a new section of program may not work but usually it can be tested. It may sound slow work to check your program after every half a dozen or so lines, but it is much easier to deal with problems one at a time than all at once at the end.

This is very good 'preventive medicine' but you may have a few problems left. These fall into two major types:

(a) problems which produce false results every time the program is run;
(b) problems which only appear occasionally, or under exceptional circumstances.

The first type are the easiest to fix, while the second type may only appear after fifty games. However, there are some helpful rules which can be applied to many of these problems:

(1) *Locate the problem*
This may be simple if the program crashes – the type of problem and line where the program crashed are given on the screen, but where the program carries on with wrong results (e.g., invader swallows up missiles, or leaves a black trail), the error might be elusive.

(2) *Find the type of problem*
Where the program crashes, a number or letter indicates the type of problem. Otherwise check – e.g., does the screen show the wrong information? Is it a problem with space? Is there an arithmetical problem?

(3) *At this stage you may be able to correct the error*
If not, check whether GOSUBs and GOTOs go to the right places. Some other common bugs are:

FOR-NEXT loops
Loops of the form
FOR F = 0 TO 20
LET A(F) = 1 (or any number, variable, etc.)
NEXT F
will not work. This is because the array A is dimensioned without A(0). So, on coming to LET A(0) = 1, the program will crash. This is a common mistake made when converting programs from other makes of computer, or by those who have upgraded from an old ROM ZX80.

This is another loop mistake:
110 FOR F = 1 TO 10
120 IF A(F)<6 THEN GOTO 140
130 NEXT F
140 LET A(F) = A(F)+6
150 PRINT A(F)
160 NEXT F

This loop will work perfectly unless A(10) = 6. In this case, A(10) will be increased by 6 and PRINTed. Remember that NEXT F means 'next F unless finished loop, in which case just carry on'. This problem can be avoided by changing the first NEXT F to
130 GOTO 160.

Numerical calculations
If a sub-routine such as
100 LET A(N+6) = A(N+6)*(BC+2)
1010 FOR G = 1 TO 5 STEP .5
1020 LET A(N+6) = A(N+6)-G*RND
1030 NEXT G
1040 LET A(N+6) = A(N+6)+INT(RND*B)
1050 RETURN
gives the wrong answer, it is difficult to see why. Pinpointing the mistake may be done by adding PRINT statements.
e.g.,
1007 PRINT "A(N+6)"; A(N+6)

1009 PRINT "N"; N
1037 PRINT "A(N+6)"; A(N+6)
1047 PRINT "B"; B
1049 PRINT "A(N+6)"; A(N+6)
also add
1050 STOP.

This shows how A(N+6) changes, and it is usually obvious where the mistake lies. For instance, you might have included a FOR-NEXT loop using FOR B = 1 TO 6, so instead of B being the cube root of 2, or whatever, it will be 6!

Note the use of line numbers ending in 7 or 9. These can easily be picked out and removed after debugging.

Space on the screen

Make sure that the screen is not overfilled by any variation of your program. This is essential in adventure games, where the screen may be filled with writing: e.g., if the hero uses a two-handed axe, rather than a sword, against a creature with a long name – say a giant caterpillar – the screen may overfill and the program gives up. Things like this can be guarded against by using CLS more frequently.

Poking in the wrong place

Always SAVE a program which uses POKE before testing it out. Then, if you accidentally muddled a few addresses and POKE flying saucers into your program, nothing is lost. Here is an example of how mis-POKEing may appear – here the black squares should have been POKEd on screen, but instead ended up on the listing.

```
 10 GOSUB 8000
100 PRINT AT Y,X;" ";AT Y,X;"█" ¦
;AT Y,X;B$
110 IF INKEY$="" THEN GOTO 100
120 LET █$=INKEY$
```

```
 200 IF A$<"1" OR A$>"8" THEN GO
TO 300
 210 LET X=X+(A$="8")+(A$="2")+(
A$="3")-(A$="5")-(A$="1")-(A$="4
")
 220 LET X=Y+(A$="5")+(A$="3")+(
A$="4")-(A$="7")-(A$="1")-(A$="2
")
 230 LET X=X+(X<0)-(X>31)
 240 LET Y=Y+(Y<0)-(Y>21)
 250 IF SP=1 THEN GOTO 100
 260 FOR F=1 TO 10
 270 NEXT F
 280 GOTO 100
 300 IF A$="0" THEN LET SP=-SP
 310 IF A$="9" THEN GOTO 400
 320 IF A$="T" THEN GOTO 500
```
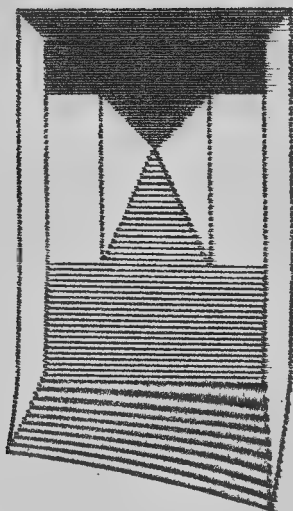
3-D Maze

This program is a game which shows the Spectrum's hi-res abilities to good effect. It occupies nearly all the memory of a 16K Spectrum, so unless you have an expanded machine, room for expansion is limited.

To use
The program forms a maze which you have to try to escape from in the shortest possible time.

When RUNning the program, you will have to wait for about a minute while the maze is randomly formed. Then you are shown your first view of the maze. You always start at a crossroads in the centre of the maze, facing north.

```
 200 IF A$<"1" OR A$>"8" THEN GO
TO 300
 210 LET X=X+(A$="8")+(A$="2")+(
A$="3")-(A$="5")-(A$="1")-(A$="4
")
 220 LET Y=Y+(A$="6")+(A$="3")+(
A$="4")-(A$="7")-(A$="1")-(A$="2
")
 230 LET X=X+(X<0)-(X>31)
 240 LET Y=Y+(Y<0)-(Y>21)
 250 IF SP=1 THEN GOTO 100
 260 FOR F=1 TO 10
 270 NEXT F
 280 GOTO 100
 300 IF A$="0" THEN LET SP=-SP
 310 IF A$="9" THEN GOTO 400
 320 IF A$="T" THEN GOTO 500
```

3-D Maze

This program is a game which shows the Spectrum's hi-res abilities to good effect. It occupies nearly all the memory of a 16K Spectrum, so unless you have an expanded machine, room for expansion is limited.

To use
The program forms a maze which you have to try to escape from in the shortest possible time.

When RUNning the program, you will have to wait for about a minute while the maze is randomly formed. Then you are shown your first view of the maze. You always start at a cross-roads in the centre of the maze, facing north.

You can move around the maze using the commands:

    on
    left
    right
    reverse

These are typed in when the computer asks 'What next?' You must press ENTER after a command.

To help you orientate yourself, by entering the command
   compass
the direction in which you are facing is shown:

      East.
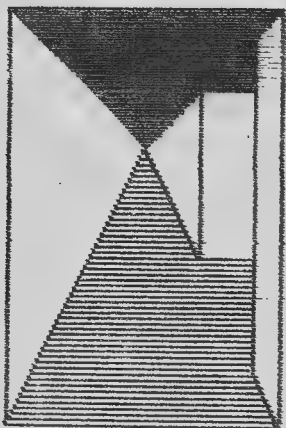
The game is played in real time, and your score at the end is given in seconds (the lower the better). You escape by exiting the maze on any border and there may be several possible exits. To see how you are getting on, entering 'time' will tell you how long you have been in the maze (in seconds).

DEAD
END

A wrong turning and precious seconds are wasted!

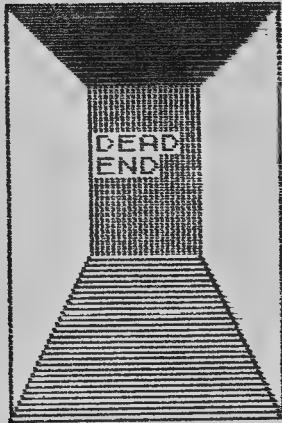If you get stuck or are convinced there is no way out of the maze (this can happen, although very infrequently) enter 'help'. This will show you an overhead diagram of most of the maze (the whole maze on a 25×25 grid will not fit on the screen). Your position is shown, and the direction is given. By frequent use of the 'help' command you can escape easily from most mazes but it's more of a challenge if 'help' is a last resort.
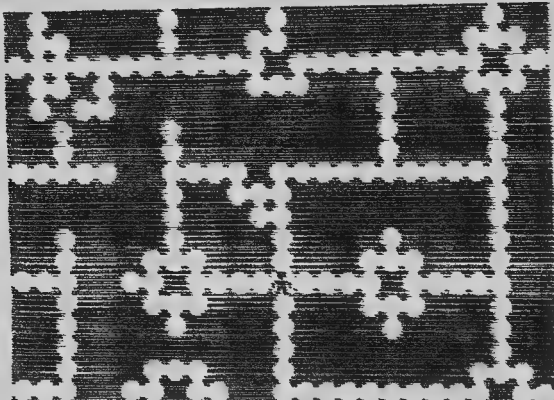
You are at the figure, facing North.

A bird's eye view makes the route to freedom clear.

Note that when using 'reverse' the effect is to move you *on* a
bit, then turn you round. So if you reverse at a crossroads you
still see a crossroads. Reversing at a T junction when facing the
T will place your back to the wall, giving a view like this:


```
East.
```
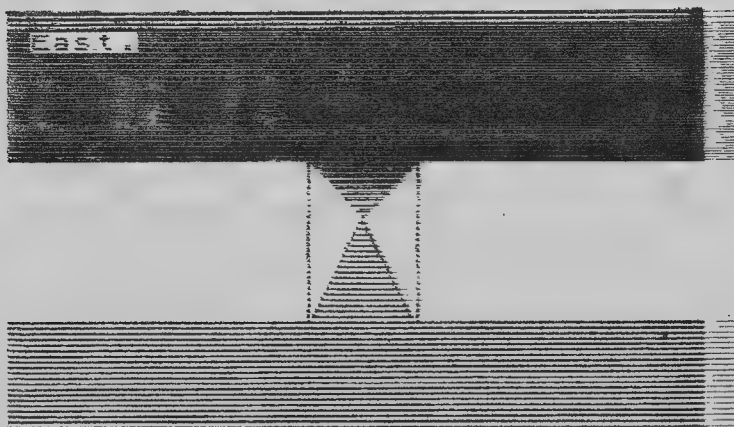
Reversing when the branch is to one side will change the branch
to the other side. Reversing at a dead-end or in a straight
corridor will show a straight corridor, and reversing in a bend
may give a view like this:

```
Time So Far
= 580 secs.
```
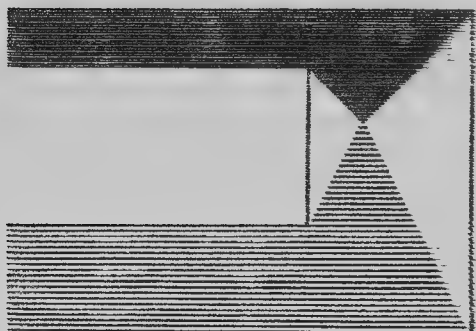
To summarize, your commands are:

> on
> reverse
> left
> right
> time
> compass

and help

If you wish to stop the program you can press BREAK while the computer is drawing a picture.

Notes on listing

The program is in three main parts. Lines 50 to 760 are mainly concerned with DRAWing the pictures on screen. Lines 1000–1760 deal with your INPUTted commands, and lines 7000–9110 contain the routines which build up the maze and variables.

10 Gosub set up maze routines.

50–130 DRAW straight passage routine. At least part of this is used for all the pictures except left and right turns as viewed from the corner and when you exit the maze.

di is the direction you are facing: 0 = north, 1 = west, 2 = south and 3 = east. af, al and ar are 0 if there is an opening, or 1 if there is a wall in front of you, to your left and to your right respectively.

In lines 80 and 110 the area directly behind you is considered, to check if you are standing against a wall.

150–180 Note use of logical operators. IF NOT al means IF al = 0. IF af means IF af > 0.

200–260 DRAW passage leading off to left.

300–350 DRAW passage leading off to right.

If passages are drawn to both sides, crossroads and T junctions can be formed.

400–420 Change left junction to left turn.

430–470 Change right junction (or crossroads) to right turn (or T junction).

500–530 Dead-end.

550–590 T junction as seen from straight wall of T.

600–660 You escape. DRAW doorway, print time taken and make noise.

700–760 This routine DRAWs left and right turns as seen from the corners.

The multiplication by al means that the routine will draw a corridor to the left when al = 0 and to the right when al = 1.

1000–1080 Enter command routine. All the drawing routines and command routines eventually end up here, where you ENTER your next command. A command is acknowledged by a short BEEP when you press ENTER.

1200–1220 Go 'on'.

1300–1330 Go 'left'.

1400–1430 Go 'right'.

Note that you cannot walk into a wall.

1500–1510 'Reverse'.

1550 'Compass'.

1600 'Time'. Uses FuNction DEFined at end of program.

1700–1760 'Help' routine. PRINTs overhead view of maze, missing off the top or bottom few lines according to your position. The use of a special character instead of a space for a corridor is due to the need for a central solid block when there is a square of four-corridor sections, as you cannot have large open spaces.

7000–7010 PRINT instructions.

8000–8660 Set up random cave system as 25 × 25 grid in blocks of 5 × 5 locations.

The POKE in line 8010 puts randomly coloured squares on the lower part of the screen, forming a simple pattern to watch while the maze is being formed.

The program stores the maze in a string array Z$, and each 5 × 5 block is chosen randomly from the selection of 10, in lines 8200–8660.

Lines 8020 to 8050 ensure that the 5 × 5 block chosen will be copied into Z$ in one of four ways, i.e., normal, reflected top to bottom, reflected left to right, or both.

Thus there are millions of ways the maze can be formed. Line 8090 puts a crossroads in the centre of the maze, so that you are unlikely to have no way out.

8800–8880 X and Y are your initial co-ordinates, i.e., in the exact centre of the maze. di is your direction, north. X$ contains the names of the directions.

8830–8850 DEFines three FuNctions similar to those shown in the manual to use the real time clock (systems variable FRAMES).

8860 sets up two user-defined graphics – a man and a corridor section.

8870 starts the internal clock. Note that this happens immediately before the game starts, so your time taken does not include the time when you are waiting for the computer to start.

9000–9030 is the DATA used to tell the computer which square is in front of you and to either side. They are used in pairs, e.g., in line 9000, 0, 1 means that when facing north, the location in front of you is $(X+0, Y+1)$.

9100 DATA for user-defined corridor cell.

9110 DATA for user-defined man.

Listing

```
   1 REM   3-D Maze
   2 REM   © S.Robert Speel

  10 GO SUB 7000
  20 INK 1: PAPER 6: BORDER 3: C
LS

  50 CLS : RESTORE 9000+10*di : R
EAD f1,f2,l1,l2,r1,r2
  60 IF x=1 OR x=25 OR y=1 OR y=
25 THEN GO TO 90
  70 LET af=VAL z$(x+f1,y+f2): L
ET al=VAL z$(x+l1,y+l2): LET ar=
VAL z$(x+r1,y+r2)
  80 IF z$(x-f1,y-f2)="1" AND al
+ar=1 THEN GO TO 700
  90 CLS : PLOT 80,0: DRAW 0,150
: DRAW 50,-50: DRAW 50,50. DRAW
0,-150: DRAW -50,100: DRAW -50,-
100
 100 IF x=1 OR x=25 OR y=1 OR y=
25 THEN GO TO 600
 110 IF z$(x-f1,y-f2)="1" AND NO
T al AND NOT ar THEN GO TO 550
```

```
 120 FOR f=100 TO 1 STEP -2: PLO
T 130-f/2,100-f: DRAW f,0: NEXT
f: IF x=1 OR x=25 OR y=1 OR y=25
 THEN GO TO 600
 130 FOR f=100 TO 1 STEP -1.6: P
LOT 130-f/2,f/2+100: DRAW f,0: N
EXT f
 150 IF NOT al THEN GO TO 200
 160 IF NOT ar THEN GO TO 300
 170 IF af THEN GO TO 500
 180 GO TO 1000

 200 PLOT 90,20: DRAW 0,120: PLO
T 90,60: DRAW 20,0: DRAW 0,60: D
RAW -20,0
 210 PLOT 90,21: DRAW INVERSE 1;
20,40
 220 FOR f=30 TO -10 STEP -2: PL
OT 90,50-f: DRAW 30-f,0: NEXT f
 230 FOR f=20 TO 1 STEP -1: PLOT
 90,140-f: DRAW f,0: NEXT f
 240 IF NOT ar THEN GO TO 300
 250 IF af THEN GO TO 400
 260 GO TO 350

 300 PLOT 170,20: DRAW 0,120: PL
OT 170,60: DRAW -20,0: DRAW 0,60
: DRAW 20,0
 310 PLOT 170,19: DRAW INVERSE 1
;-20,40
 320 FOR f=30 TO -10 STEP -2: PL
OT 170,f-50: DRAW f-30,0: NEXT f
 330 FOR f=20 TO 1 STEP -1: PLOT
 170,140-f: DRAW -f,0: NEXT f
 340 IF af THEN GO TO 450
 350 GO TO 1000

 400 FOR f=61 TO 119: PLOT INVER
SE 1;110,f: DRAW INVERSE 1;60,0:
 NEXT f
 410 PLOT 150,60: DRAW 0,60
 420 GO TO 1000
 450 FOR f=61 TO 119: PLOT INVER
SE 1;110,f: DRAW INVERSE 1;59,0:
 NEXT f
 460 IF al THEN PLOT 110,60: DRA
W 0,60
 470 GO TO 1000

 500 FOR f=60 TO 120: PLOT INVER
SE 1;110,f: DRAW INVERSE 1;59,0:
 NEXT f
 510 FOR f=110 TO 150 STEP 2: PL
OT f,60: DRAW 0,60: NEXT f
```

```
  520 PRINT AT 9,14;"DEAD";AT 10,
14;"END"
  530 GO TO 1000

  550 FOR f=7 TO 14: PRINT AT f,8
;"    ";AT f,21;"    ": NEXT f: F
OR f=14 TO 21: PRINT AT f,0; PAP
ER 6-(f>18)*3;"    ": NEXT f
  560 FOR f=24 TO 63 STEP 2: PLOT
 0,f: DRAW OVER 1;255,0: NEXT f
  570 PRINT AT 0,0;: FOR f=0 TO 6
: PRINT "          ": NEXT f
  580 FOR f=0 TO 1: PLOT 110+f*40
,63: DRAW 0,57: NEXT f: FOR f=60
 TO 100 STEP 2: PLOT 80+f/2,f: D
RAW 100-f,0: NEXT f
  590 FOR f=100 TO 120: PLOT 230-
f,f: DRAW f*2-200,0: NEXT f: GO
TO 1000
  600 BORDER 4: FOR f=10 TO 0 STE
P -10: PLOT 100-f,20: DRAW 0,80+
f: DRAW 30+f,40+f: DRAW 30+f,-40
-f: DRAW 0,-80-f: NEXT f
  610 FOR f=20 TO 100: PLOT 100,f
: DRAW 60,0: NEXT f
  620 FOR f=100 TO 140: PLOT 25+f
*3/4,f: DRAW 212-f*19/12.5,0: NE
XT f
  630 FOR f=1 TO 30: PLOT OVER 1;
 PAPER 6;100+RND*60,20+RND*80: N
EXT f
  640 PRINT AT 0,0;" You have rea
ched the exit."'"You took"'FN c(
);" Seconds."
  650 FOR f=10 TO 67: BEEP .1,f-5
0: NEXT f: BEEP 2,18
  660 STOP

  700 FOR f=19 TO 21: PRINT AT f,
0; PAPER 3;"          ": NEXT f
  710 FOR f=24 TO 63 STEP 2: PLOT
 al*(f/2+80),f: DRAW 180-f/2-al*
5,0: NEXT f
  720 FOR f=60 TO 100 STEP 2: PLO
T 80+f/2,f: DRAW 100-f,0: NEXT f
  730 FOR f=100 TO 120: PLOT 230-
f,f: DRAW f*2-200,0: NEXT f
  740 FOR f=120 TO 140: PLOT al*(
230-f),f: DRAW 30+f-al*5,0: NEXT
 f
```

```
 750 PLOT 110+al*40,63: DRAW 0,5
6: PLOT 170-al*80,24: DRAW 0,116
 760 GO TO 1000
1000 INPUT "What next? ";a$: BEE
P 1,20
1010 IF a$="on" THEN GO TO 1200
1020 IF a$="left" THEN GO TO 130
0
1030 IF a$="right" THEN GO TO 14
00
1040 IF a$="reverse" THEN GO TO
1500
1050 IF a$="compass" THEN GO TO
1550
1060 IF a$="time" THEN GO TO 160
0
1070 IF a$="help" THEN GO TO 170
0
1080 GO TO 1000

1200 IF af THEN GO TO 1000
1210 LET x=x+f1: LET y=y+f2
1220 GO TO 50
1300 IF al THEN GO TO 1000
1310 LET x=x+l1: LET y=y+l2
1320 LET di=di+1: IF di>3 THEN L
ET di=0
1330 GO TO 50

1400 IF ar THEN GO TO 1000
1410 LET x=x+r1: LET y=y+r2
1420 LET di=di-1: IF di<0 THEN L
ET di=3
1430 GO TO 50

1500 LET di=di+2: IF di>3 THEN L
ET di=di-4
1510 GO TO 50

1550 PRINT AT 1,1;x$(di*5+1 TO d
i*5+5): GO TO 1000

1600 PRINT AT 0,15;"Time So Far"
;TAB 15;" = ";FN t();" secs.": G
O TO 1000

1700 LET fp=25-(y<13)*7
1710 CLS : FOR f=fp TO fp-17 STE
P -1: FOR g=1 TO 25: LET g$=CHR$
 144: IF z$(g,f)="1" THEN LET g$
="▓"
1720 IF x=g AND y=f THEN PRINT I
NK 0; FLASH 1;CHR$ 145;: GO TO 1
740
```

```
1730 PRINT g$;
1740 NEXT g: PRINT : NEXT f
1750 PRINT "You are at the figur
e, facing  ";x$(di*5+1 TO di*5+5
);"."
1760 GO TO 1000

7000 PRINT "Corridors"''"Command
s are:"''"on"''"reverse"''"left"''"
right"''"compass"''"time"''"and hel
p."
7010 PRINT "Your aim is to exit
the caves in the shortest possib
le time."

8000 PRINT "Please Wait": RANDOM
IZE : DIM z$(25,25): DIM y$(5,5)
8010 FOR f=1 TO 25 STEP 5: FOR g
=1 TO 25 STEP 5: POKE 23000+RND*
295,RND*255
8020 LET ax=1: IF RND<.5 THEN LE
T ax=5
8030 LET ay=1: IF ax=5 THEN LET
ay=-1
8040 LET bx=1: IF RND<.5 THEN LE
T bx=5
8050 LET by=1: IF bx=5 THEN LET
by=-1
8060 GO SUB 8200+INT (RND*10)*50
8070 FOR j=0 TO 4: FOR k=0 TO 4:
 LET z$(f+j,g+k)=y$(j*ay+ax,k*by
+bx): NEXT k: NEXT j
8080 NEXT g: NEXT f: GO SUB 8200
8090 FOR j=1 TO 5: LET z$(10+j,1
1 TO 15)=y$(j): NEXT j: GO TO 88
00

8200 FOR h=1 TO 5: LET y$(h)="11
011": NEXT h
8210 LET y$(3)="00000": RETURN

8250 FOR h=1 TO 5: LET y$(h)="11
011": NEXT h
8260 LET y$(3)="11000": RETURN

8300 LET y$(1)="11111": LET y$(2
)=y$(1)
8310 LET y$(4)="11011": LET y$(5
)=y$(4)
8320 LET y$(3)="00000": RETURN

8350 LET y$(1)="11111": LET y$(2
)=y$(1)
8360 LET y$(4)="11011": LET y$(5
)=y$(4)
```

```
8370 LET y$(3)="00011": RETURN

8400 LET y$(1)="11011": LET y$(5
)=y$(1)
8410 LET y$(2)="10001": LET y$(4
)=y$(2)
8420 LET y$(3)="00100": RETURN

8450 LET y$(1)="11011": LET y$(5
)=y$(1)
8460 LET y$(2)="10101": LET y$(3
)="00100"
8470 LET y$(4)="10111": RETURN


8500 LET y$(1)="11011": LET y$(2
)="00101"
8510 LET y$(3)="10100": LET y$(4
)="10001"
8520 LET y$(5)="11011": RETURN

8550 LET y$(1)="11000": LET y$(2
)="11010"
8560 LET y$(3)="10001": LET y$(4
)="00000"
8570 LET y$(5)="11011": RETURN

8600 LET y$(1)="11011": LET y$(5
)=y$(1)
8610 LET y$(2)="10011": LET y$(3
)="10100"
8620 LET y$(4)="10001": RETURN

8650 LET y$(1)="11001": LET y$(2
)="11100"
8660 LET y$(5)="11011": RETURN

8800 LET x=13: LET y=13
8810 LET di=0
8820 LET x$="NorthWest SouthEast
."
8830 DEF FN a()=INT ((PEEK 23672
+256*PEEK 23673+65536*PEEK 23674
)/50)
8840 DEF FN b(x,y)=(x+y+ABS (x-y
))/2
8850 DEF FN c()=FN b(FN a(),FN a
())
8860 RESTORE 9100: FOR f=0 TO 1:
 FOR g=0 TO 7: READ a: POKE USR
CHR$ (144+f)+g,a: NEXT g: NEXT f
8870 FOR f=23674 TO 23672 STEP -
1: POKE f,0: NEXT f
8880 RETURN
```

```
9000 DATA 0,1,-1,0,1,0
9010 DATA -1,0,0,-1,0,1
9020 DATA 0,-1,1,0,-1,0
9030 DATA 1,0,0,1,0,-1

9100 DATA 195,129,0,0,0,0,129,19
5
9110 DATA 56,56,16,124,166,186,4
0,108
```

Boat Race

To use

In this Spectrum program you have to bet on a race between four boats. You start with £100 and you need £500 to win. You may bet any amount on each race.

```
Boat race
```
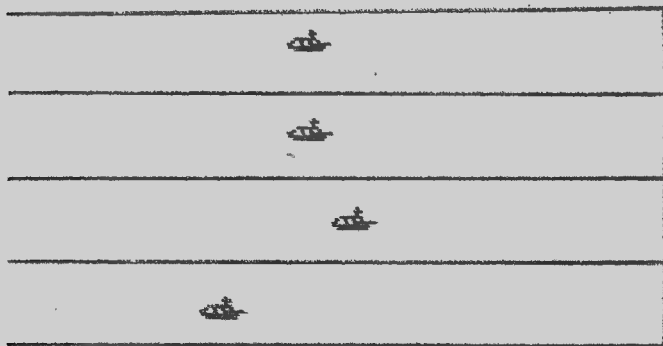


```
You bet £15 on black
```

Notes on listing

10 A$ holds the colours of the boats.

20 Define 'boat graphic' from data.

160–650 The actual race.

Lines 110–130 draw the course which the boats take, and the finishing line.

200 This gives the engine noise.

210 For each of the four boats, print boat f at correct position.

220–230 X(f) contains the boat's position, and this is increased by a random amount. If the race has been won, goto 500.

240–250 Repeat for all four boats and then repeat until winner found.

500–530 Race over, winner announced, short victory tune.

600–650 If you chose the winner goto 700. Otherwise deduct your bet from your money, check if you have lost, and start new race procedure.

700 You chose the winner, and get three times your original stake money as winnings.

800 Print your current cash and goto 'start new race'.

900–940 You win, short tune played, game ends.

1000 You start with £100.

1020–1050 List alternative boats on which bets can be placed.

1100–1130 Input choice of boat and amount of bet, make sure you have the money and start the race.

8000–8010 Back and front halves of boat, respectively.

Listing

```
  1 REM  BOATRACE
  2 REM      Copyright 1982
             S.Robert Speel

 10 LET a$="blackblue red   mauv
e"
 20 FOR f=1 TO 2: FOR g=0 TO 7:
READ a: POKE USR CHR$ (143+f)+g
,a: NEXT g: NEXT f
 30 RANDOMIZE
 50 INK 0: PRINT "Boat race"
100 GO TO 1000

110 FOR f=1 TO 5
120 PLOT 10,f*30: DRAW 240,0: N
EXT f
130 DRAW 0,-120
140 PRINT "Boat race"
150 DIM x(4)
160 PRINT AT 20,1;"You bet £";b
et;" on ";a$(ch*5-4 TO ch*5)

200 FOR F=1 TO 4: BEEP .005,2:
BEEP .005,-5
```

```
 210 INK (f-1): PRINT AT f*4,x(f
);"  ";CHR$ 144;CHR$ 145
 220 LET x(f)=x(f)+RND*1.5
 230 IF x(f)>28 THEN GO TO 500
 240 NEXT f
 250 GO TO 200


 500 INK 0: PRINT AT 20,1;"Race
over.";
 510 PRINT "The ";a$((f-1)*5+1 T
O f*5);" boat wins."
 520 FOR g=1 TO 24: BEEP .2,g-5:
NEXT g
 530 BEEP 1,20


 600 CLS
 610 IF ch=f THEN GO TO 700
 620 PRINT "you lose your bet...
"
 630 LET cash=cash-bet: IF cash>
0 THEN GO TO 800
 640 PRINT "you have no money le
ft-you lose."
 650 STOP
 700 PRINT "Your bet has paid of
f!!!!!"
 710 PRINT "You win £";3*bet
 720 LET cash=cash+3*bet
 730 IF cash>500 THEN GO TO 900


 800 PRINT "you now have £";cash
 810 GO TO 1020


 900 PRINT "you have now amassed
£";cash
 910 PRINT "and can afford to bu
y your own  speedboat."
 920 LET c$="967676545432111"
 930 FOR f=1 TO 15. BEEP .2,VAL
c$(f): NEXT f
 940 STOP




1000 LET cash=100
1010 INK 0: PRINT "you have £100
"
1020 PRINT "which boat do you be
t on?"
1030 FOR f=1 TO 4
1040 PRINT f;"...";a$(f*5-4 TO f
*5)
1050 NEXT f
```

```
1100 INPUT ch
1110 PRINT "how much do you bet?
"
1120 INPUT bet: IF bet>cash THEN
 LET bet=cash
1130 CLS : GO TO 110

8000 DATA 0,0,0,31,36,196,127,63
8010 DATA 64,224,64,224,144,255,
252,248
```

Sheepdog

You control a sheepdog and muster a small flock of four sheep through a gate into the next field. Your aim is to do this in as short a time as possible. The sheepdog is moved using the 5, 6, 7 and 8 keys. The sheep wander around at random, but if the sheepdog comes too near, they will run away. Once through the gate to the right of the screen, a sheep will not come back.

When you start, you may find it difficult to get even one sheep through the gate, but with a little practice you will improve. The way to get a really fast time is to keep the sheep in a flock and get all four to go through the gate together. If you split up the sheep and chase them through one by one, you will not get a very good score.
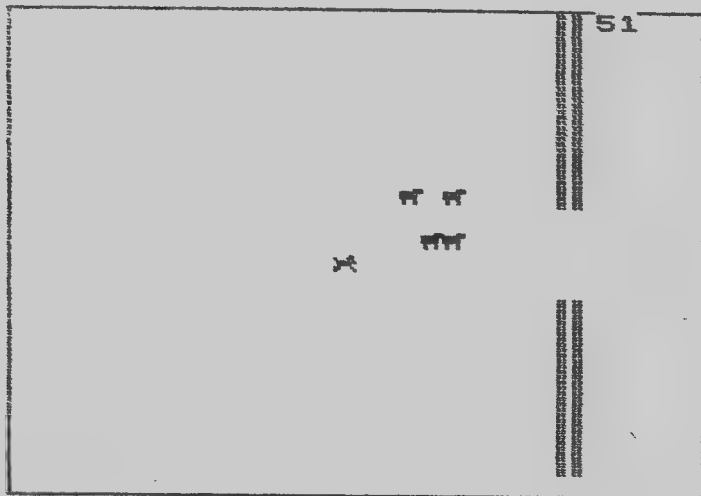
A clever sheepdog, keeping the sheep together.
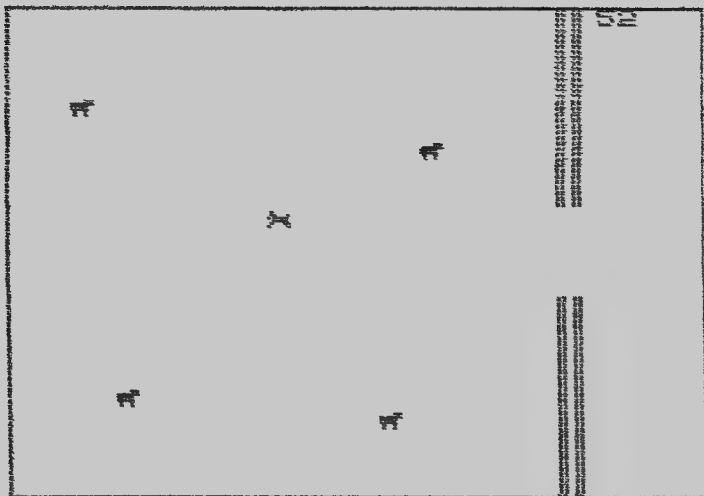
The flock has split – what every sheepdog dreads.

Notes on listing
This game is designed for expanding and a few parts make little sense in this shorter version, notably string array A$.

10 Gosub set up user-defined graphics and variables.
1000–2520 Main loop.
　　Lines 1000–2020 are repeated once for each sheep.
1010–1020 sy and sx are the y and x movements of a sheep. Initially, they are set according to whether the sheepdog is within three character spaces away.
1030 Random factor to make sheep wander around in a typical sheep-like manner.
1040–1050 Make sure sheep's move will not take it offscreen, or over the fence. Note it is possible (if you are very lucky indeed) for a sheep to jump the fence, but in general they will only go through the gate. Note these lines also keep a sheep in the other field once it has passed through the gate.
1060–1080 Change sheep's position co-ordinates by adding sx and sy. PRINT space at sheep's old position and user-defined sheep at new position.

1090 Check for game finished (all sheep in other field). If so, goto end of game.

1500 Print sheepdog, facing left or right. Note dog usually faces right, as this is the way the sheep should be herded.

2000–2010 Change x and y co-ords of dog according to key pressed. Note that as this happens inside the f loop, once for each sheep, the dog can run four times as fast as the sheep. This gives enough manoeuvrability for the dog to be able to keep the sheep in a flock, if you are skilled enough.

2020 Repeat for all four sheep.

2030 Increment go counter.

2500–2520 Print rough estimates of time taken. Every 20 goes, reDRAW fence as sheep often dent it.
Repeat cycle until game finished.

4000–4200 End of game.

4010 finds exact time taken.

4020 gives short 'tune'.

4100–4110 deals with the highscore.

4120–4200 offers another game.

8000–8520 used in each game. Array a holds the sheep's co-ords, and di is the direction the dog is facing in: 0 = right, 1 = left.

8200–8210 sets up a user-defined function to calculate time, and POKEs the internal clock to zero.

8500–8510 Set colours and draw fence. Fence routine often used (every twenty goes) to keep fence intact.

9000–9010 User-defined sheep and sheepdog.

Listing

```
   1 REM Sheepdog
   2 REM  © S.Robert Speel 1982

  10 GO SUB 8000

1000 FOR f=1 TO 7 STEP 2
1010 LET sy=SGN (a(f)-y) AND ABS
     (a(f+1)-x)<4 AND ABS (a(f)-y)<4
1020 LET sx=SGN (a(f+1)-x) AND A
BS (a(f)-y)<4 AND ABS (a(f+1)-x)
<4
```

```
1030 LET sx=sx+INT (RND*3)/2-.5:
 LET sy=sy+INT (RND*3)/2-.5
1040 LET sx=sx+(a(f+1)(1)*2-(a(f
+1)>29)*2-(a(f+1)>22 AND a(f+1)<
27 AND (a(f)<9 OR a(f)>12))+(a(f
+1)>25 AND a(f+1)<27)*2
1050 LET sy=sy+(a(f)<2)*2-(a(f)>
19)*2
1060 LET a(f)=a(f)+sy: LET a(f+1
)=a(f+1)+sx
1070 PRINT AT a(f)-sy,a(f+1)-sx;
" "
1080 PRINT INK VAL a$(f);AT a(f)
,a(f+1);CHR$ 146
1090 IF a(2)>25 AND a(4)>25 AND
a(6)>25 AND a(8)>25 THEN GO TO 4
000

1500 PRINT AT y-y1,x-x1;" "; INK
 0;AT y,x;CHR$ (144+di)

2000 LET x1=(INKEY$="8")-(INKEY$
="5")+(x<2)-(x>23): LET y1=(INKE
Y$="6")-(INKEY$="7")+(y<1)-(y>19
): LET di=(INKEY$="5")
2010 LET x=x+x1: LET y=y+y1
2020 NEXT f
2030 LET go=go+1

2500 PRINT INK 0;AT 0,27;INT (go
*1.5)
2510 IF go/20=INT (go/20) THEN G
O SUB 6510
2520 GO TO 1000
4000 INK 0: PRINT AT 10,2;"You h
ave penned all";AT 11,2;"the she
ep."
4010 LET sc=FN c(): PRINT AT 13,
2;"You took ";sc;" Seconds."
4020 INK 0: FOR f=-25 TO 25: BEE
P .1,f: BEEP .1,25-f: NEXT f

4100 IF hs>sc THEN LET hs=sc: PR
INT AT 15,1;"Well done - a new r
ecord!"
4110 PRINT AT 17,1;"Highscore =
";hs
4120 PRINT AT 20,3;"Another game
? (y/n)"
4130 IF INKEY$="y" THEN GO TO 42
00
4140 IF INKEY$="n" THEN PAPER 7:
 BORDER 7: CLS : STOP
4150 GO TO 4130
```

```
4200 GO SUB 8100: GO TO 1000

5000 FOR f=0 TO 2: FOR g=0 TO 7:
 READ a: POKE USR CHR$ (144+f)+g
,a: NEXT g: NEXT f
6010 LET hs=500

8100 DIM a(8): DIM a$(8): FOR f=
1 TO 7 STEP 2: LET a(f)=10+f: LE
T a(f+1)=5
8110 LET a$(f)="6": LET a$(f+1)=
"1": NEXT f
8120 LET x=1: LET y=1: LET di=0
8130 LET x1=0: LET y1=0: LET go=
0

8200 DEF FN a(m,n)=(m+n+ABS (m-n
))/2: DEF FN b()=(PEEK 23672+256
*PEEK 23673+65536*PEEK 23674)/50
: DEF FN c()=FN a(FN b(),FN b())
8210 POKE 23672,0: POKE 23673,0:
 POKE 23674,0

8500 PAPER 4: BORDER 6: CLS
8510 INK 2: FOR f=1 TO 4: PLOT 2
00+f*2+(f>2)*2,0: DRAW 0,72: DRA
W INK 4;0,32: DRAW 0,71: NEXT f
8520 RETURN

9000 DATA 0,2,134,125,124,194,12
9,0,0,64,97,190,62,67,129,0
9010 DATA 0,7,255,252,252,68,68,
0
```

62

You have penned all
the sheep.

You took 91 Seconds.

Well done - a new record!

Highscore = 91

   Another game? (y/n)

Sheepdog Champion

When you have become proficient at Sheepdog you can try this
more advanced version.

To use
The aim is the same as for Sheepdog, but this time there is more
skill needed. You may have wondered why the sheep are yellow
rather than white. The answer is that they are dirty. You will
have to drive the sheep into the left end of the sheep-dip at the
top left of the screen. They will come out clean and white. For
each sheep that you fail to dip, you add 25 seconds to your end
score.

There are bushes dotted around the right-hand side of the
field, and the sheep may stop at these to graze on the leaves.
Neither sheep nor sheepdog can actually go into a bush,
although the sheep can jump over them. Now that you are so
expert, you can naturally handle six sheep rather than the
original four.
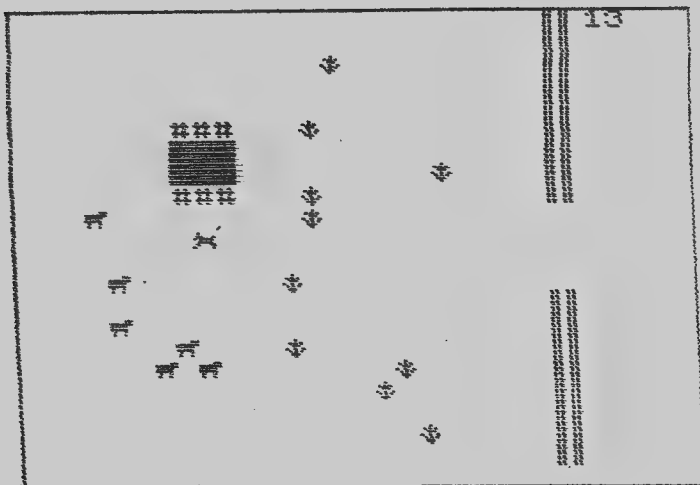
Notes on listing

IMPORTANT *This is not a complete listing.* Type it in exactly as shown, then MERGE with Sheepdog. The complete Sheepdog Champion can then be SAVEd normally. Note that this listing must be MERGEd on top of Sheepdog, *not* the other way round (i.e., type this in, SAVE it, LOAD Sheepdog, and MERGE this listing).

1000 deals with sheep-dip. Sheep's colour changed, bleat noise, move sheep one onwards and go to 'change sheep's co-ords'.

1030–2000 Some minor changes and moving lines around.

1090 checks if six sheep rather than four are through the gate.

2010 Check if a tree is in the character-square where a sheep would otherwise be printed. Trees are BRIGHT so have large ATTRibutes.

2500 The rough 'time taken' has to be altered.

4030–4060 Check how many sheep have been through sheep-dip. Make your score worse by 25 for each un-dipped dirty sheep.

7000–7050 Rules have been added. These can be expanded as desired.

8000 There is one extra graphic to be defined.

8100 Arrays for six sheep instead of four.

8500 PRINT 10 bushes at random in right hand part of field. Note that they are BRIGHT for easy detection by ATTR.

8520 PRINT sheep-dip.

9020 DATA for user-defined bush.

Listing (LOAD SHEEPDOG and MERGE this listing)

```
   1 REM Sheepdog Champion

  10 INK 0: PAPER 5: BORDER 1: C
LS : GO SUB 7000

1000 FOR f=1 TO 11 STEP 2: IF (a
(f)=6 OR a(f)=7) AND a(f+1)>5 AN
D a(f+1)<12 THEN LET sx=1: LET a
d=1: LET sy=0: LET a$(f)="7": LE
T a$(f+1)="0": FOR g=1 TO 5: BEE
P .01,10-RND*9: NEXT g: GO TO 10
60
```

```
1030 LET sx=sx+INT (RND*3)/2-.5:
 LET sy=sy+INT (RND*3)/2-.5: IF
(a(f)=6 OR a(f)=7) AND a(f+1)<14
 AND a(f+1)>10 THEN LET sx=ABS s
x
1050 LET sy=sy+(a(f)<2)*2-(a(f)>
19)*2+(a(f+1)>5 AND a(f+1)<9 AND
 a(f)=9)
1060 IF ATTR (a(f)+sy,a(f+1)+sx)
=96 THEN LET sx=0: LET sy=0

1070 LET a(f)=a(f)+sy: LET a(f+1
)=a(f+1)+sx: PRINT AT a(f)-sy,a(
f+1)-sx;" "
1080 PRINT INK VAL a$(f);AT a(f)
,a(f+1);CHR$ 146: IF ad THEN GO
SUB 8520: LET ad=0
1090 IF a(2)>25 AND a(4)>25 AND
a(6)>25 AND a(8)>25 AND a(10)>25
 AND a(12)>25 THEN GO TO 4000
2000 LET x1=(INKEY$="8")-(INKEY$
="5")+(x<1)-(x>23): LET y1=(INKE
Y$="6")-(INKEY$="7")+(y<1)-(y>20
): LET di=(INKEY$="5")
2010 IF ATTR (y+y1,x+x1)=96 THEN
 LET x1=0: LET y1=0
2020 LET x=x+x1: LET y=y+y1
2030 NEXT f: LET go=go+1

2500 PRINT INK 0;AT 0,27;INT (go
*3)
2510 IF go/10=INT (go/10) THEN G
O SUB 8510

4030 LET sd=0: FOR f=1 TO 6: LET
 sd=sd+VAL a$(f*2): NEXT f
4040 PRINT AT 15,2;6-sd;" Sheep
were cleaned"
4050 PRINT AT 17,3;"Penalty = ";
sd*25
4060 LET sc=sc+25*sd

4100 IF hs>sc THEN LET hs=sc: PR
INT AT 18,1;"Well done - a new r
ecord!"
4110 PRINT AT 19,1;"Top score =
";hs
4120 PRINT AT 21,3;"Another game
? (y/n)"

7000 PRINT " Sheepdog";AT 2,1;"
In this game you control a"
```

```
7010 PRINT "sheepdog using keys
5,6,7 and 8. Your aim is to herd
 the 6 sheepinto the field on th
e right."
7020 PRINT " However, the sheep
are dirty   and should first be
driven "
7030 PRINT "through the railed s
heepdip. If you fail to dip the
sheep you   will be penalised."
7040 PRINT " The aim is to get t
he flock    dipped and penned in
 as short a time as possible."
7050 PRINT ''" (press ENTER to s
tart.)": INPUT z$

8000 FOR f=0 TO 3: FOR g=0 TO 7:
 READ a: POKE USR CHR$ (144+f)+g
,a: NEXT g: NEXT f
8100 DIM a(12): DIM a$(12): FOR
f=1 TO 11 STEP 2: LET a(f)=8+f:
LET a(f+1)=5
8140 LET ad=0

8210 POKE 23672,0: POKE 23673,0:
 POKE 23674,0

8500 PAPER 4: BORDER 6: CLS : FO
R f=1 TO 10: LET rs=INT (RND*20)
: LET pr=15+INT (RND*8): PRINT I
NK 0; BRIGHT 1;AT rs,pr;CHR$ 147
: NEXT f
8520 PRINT PAPER 5; INK 1;AT 5,8
;"###";AT 6,8;"   ";AT 7,8;"
;AT 8,8;"###": RETURN

9020 DATA 0,16,84,56,146,84,56,1
6
```

Knight Fight

To use

This game involves a fight between two knights.

You control a white knight, and the computer has a black knight. At the start, you have to select armour and weapons for your knight. You have 100 gold coins to spend, which means you cannot have the heaviest armour and lance and the best weapons too.

A mace is the strongest attacking weapon, a sword the weakest. However, a sword is useful in defence as well as attack, whereas an axe or mace is not.

Once you have chosen your weapons, the enemy's choice is announced. The fight then begins.

At first, the knights joust on horseback until one knight is knocked off his horse. Apart from your choice of lance and armour, you have no control over the jousting. Once one knight is unhorsed, the other will dismount and the fight continues on foot.
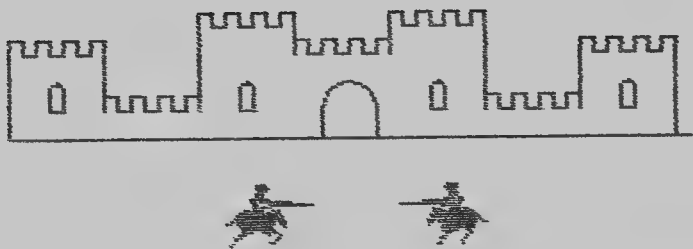
You attack the opponent knight by moving near to him using the keys 5 and 8 to move. You 'hit' by pressing one of the keys 1, 2, 3 or 4 – the number determining the strength of hit. Each

time you hit, your strength, shown at the bottom left of the screen, goes down by that number. If your strength reaches zero, you die. By moving away from the enemy, you can gradually recover your strength up to your maximum.
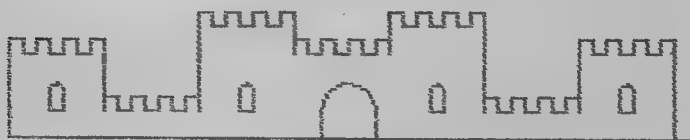
A successful hit on the enemy lowers his strength by one *permanently*. The enemy strikes you in exactly the same way. This means that fighting consists of rushing up to your opponent, hitting at him a few times, then retiring to recover from your exhaustion. It is essential that you keep an eye on your own strength to know when to retreat. Gradually your maximum strength will be depleted (and so, hopefully, will your opponent's) until one or the other, with a strength of three or less, cannot strike properly. Death for the weaker usually follows quickly.

Notes on listing
This is rather a long game which occupies nearly all of the basic 16K Spectrum. Due to the large numbers of user-defined graphics, the use of CHR$ 144, etc. has been dropped, and graphics characters are used in the listings. All capital letters inside quotes should be graphic letters, i.e., you must go into graphics mode, *then* press the letter. If you forget one or two,

You die.   8

you may see knights charging around on the backs of ABC
creatures. If this occurs, BREAK the program, find the letter,
and replace it with a graphic letter.

10 Gosub start routine, buy weapons and set up user-defined
graphics.

100–110 Basic 'charge on horseback' routine. Graphics ABC
and LKJ are the horses, DEF/GHI and ONM/RQP are
alternate sets of legs to give a galloping effect. In lines 100
and 130, the underline character is used (symbol shift 0) as
the point of a lance, US and TU are the rest of the lance and
the knight.

160 Gives a simulation of the sound of horses' hooves, which
sounds very good when amplified.

170 Deals with a possible crash between the two knights.

**180** Checks if the knights are about to go offscreen.

200–910 Copes with what happens when the knights meet on
horseback.
200–240 is the actual hit, strengths of opponents being
calculated to take in armour, lance-type and shields.
500–540 shows the left knight falling off.
600–840 shows the right knight falling off.

800–840 checks if a knight has fallen off at the end of a run.

900–910 deals with turning at the end of a run when neither knight is knocked off his horse.

1000–1040 Turns horses round at end of run when one man is down; and dismounts other knight.

1100–1430 Foot-fighting loop.

1100 PRINT two knights with correct weapons.

1120 moves your knight.

1130 moves opponent's knight, including automatic retreat when strength below 3.

1200–1260 Your knight hits opponent if in range. Checks if you die from exhaustion or enemy dies from wounds.

1300–1320 Enemy hits you. Your hits and your opponent's depend for their success on the strength of hit (1–4), the weapon's attack value (1–3), its defence value if it is a sword, armour and shields, if any, together with a random attack value. When all these are added, the hit is judged successful or unsuccessful.

1330–1340 If your current strength is larger than your maximum strength, then let current strength = maximum strength.

1400–1430 If knights are a distance apart, then they can gradually recover to their current maximum strength.

4000–4010 You die.

4500–4510 Enemy dies.

7000–7280 Start of game. You choose your weapons and armour, from list giving prices. Note improper entries ignored.

7320–7460 Enemy chooses weapons. Either enemy chooses to have a strong attack or a strong defence.

8000–8500 DRAW castle in background and start fight.

9000–9130 DATA for user-defined graphics used for knights, horses and weapons.
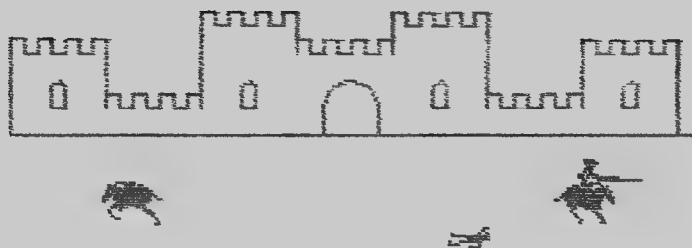
Listing

```
   1 REM Knight Fight
   2 REM  © S.Robert Speel 1982

  10 PAPER 4: BORDER 4: CLS : GO
SUB 7000

 100 INK ec: IF eh=1 THEN PRINT
AT y-1,x-1;"_US "
 110 PRINT AT y,x;"ABC ": IF x/2
=INT (x/2) THEN PRINT AT y+1,x;"
DEF "
 120 IF x/2<>INT (x/2) THEN PRIN
T AT y+1,x;"GHI "
 130 INK ac: IF ah=1 THEN PRINT
AT y-1,29-x;" TU_"
 140 PRINT AT y,28-x;" LKJ": IF
x/2=INT (x/2) THEN PRINT AT y+1,
28-x;" ONM"
 150 IF x/2<>INT (x/2) THEN PRIN
T AT y+1,28-x;" RQP"
 160 LET x=x-1: BEEP .003,10: PA
USE 2: BEEP .003,5: PAUSE 3: BEE
P .003,0: PAUSE 5
 170 IF x=15 AND ec=7 THEN GO TO
200
 180 IF x<2 THEN GO TO 800
 190 GO TO 100

 200 LET ad=le+INT (RND*6)-ar-(a
r=2)-INT (RND*6): LET ed=la+INT
(RND*4)-er-(er=2)-INT (RND*4)
 210 IF ad<=0 AND ed<=0 THEN GO
TO 100
 220 IF ad>0 AND ed>0 THEN GO TO
500+100*(ad>ed)
 230 IF ad>0 THEN GO TO 600
 240 GO TO 500
```

```
 500 PRINT AT y-1,x-4;" T/\": BE
EP .1,50: BEEP .1,30
 510 PRINT AT y-1,x-4;"    ": BEEP
 .1,20: BEEP .1,10
 520 RESTORE 9100: FOR f=0 TO 1:
 FOR g=0 TO 7: READ a: POKE USR
CHR$ 163+g,a: NEXT g: PRINT AT y
+1,x-6+f;CHR$ 163;: NEXT f
 530 PRINT "          ";AT y,x+3;"
 ";AT y,28-x;" "
 540 LET x=x-1: LET ex=x+9: LET
y=y-1: LET ah=0: LET em=em-INT (
RND*6)-1: LET se=se-6: GO TO 100


 600 PRINT AT y-1,x;"/\5": BEEP
.1,50: BEEP .1,30
 610 PRINT AT y-1,x;"          ": B
EEP .1,50: BEEP .1,30
 620 INK 7: RESTORE 9100: FOR f=
0 TO 1: FOR g=0 TO 7: READ a: PO
KE USR CHR$ 162+g,a: NEXT g: PRI
NT AT y+1,x+4+f;CHR$ 162;: NEXT
f
 630 PRINT AT y+1,x-4;"          "
;AT y,x+3;" ";AT y,28-x;" "
 640 LET x=x-1: LET ex=x+9: LET
y=y-1: LET eh=0: LET am=am-INT (
RND*4): LET sa=sa-5: GO TO 100


 800 IF eh=1 AND ah=1 THEN GO TO
 900
 810 LET x=20: LET ex=10: LET y=
y+1
 820 IF eh=1 THEN LET x=5
 830 IF ah=1 THEN LET ex=25
 840 GO TO 1000


 900 PRINT AT y-1,x;"       ";AT y-
1,28-x;"        "
 910 LET x=26: LET y=10: LET dc=
ac:: LET ac=ec: LET ec=dc: GO TO
 100


1000 PRINT AT y-1,0;" ";TAB 31;"
 "; INK 7;AT y,0;" LKJ"; INK 0;T
AB 28;" ABC"; INK 7;AT y+1,0;" O
NM"; INK 0;AT y+1,28;" GHI"
1010 FOR f=1 TO 2: PRINT AT y-f,
0;"      ";AT y-f,27;"      ": NEXT
f
1020 RESTORE 9060: FOR f=1 TO 2:
 FOR g=0 TO 7: READ a: POKE USR
CHR$ (143+f)+g,a: NEXT g: NEXT f
```

```
1030 RESTORE 9100: FOR f=3 TO 12
: FOR g=0 TO 7: READ a: POKE USR
 CHR$ (143+f)+g,a: NEXT g: NEXT
f
1040 PRINT AT y+1,5; "

1100 PRINT INK 0;AT y,ex;v$;"A "
;AT y+1,ex;" E "; INK 7;AT y,x-1
;" B";w$;AT y+1,x-1;" F "
1110 IF ex>x+2 THEN PRINT AT y,e
x-1;" ";AT y,x+3;" "
1120 LET x=x+(INKEY$="8")-(INKEY
$="5")+(x<6)-(x>25): LET x=x-(x>
ex-2)
1130 LET ex=ex+SGN (-(ex>x+1 AND
 RND<.5)+(ex<5)-(ex>25)+(RND<.3)
+(se<3 AND ex<25)): IF ex-x>2 TH
EN GO TO 1400
1200 LET a$=INKEY$: IF a$<"1" OR
 a$>"4" THEN LET a$="0"
1210 LET ez=INT (RND*4)+1: IF se
<ez THEN LET ez=se-1
1220 IF se<3 THEN LET ez=0
1230 IF a$="0" THEN GO TO 1300
1240 LET sa=sa-VAL a$: IF sa<1 T
HEN GO TO 4000
1250 IF VAL a$+wa+(RND*7)+1>ez+e
s+er-1+(we=1) THEN LET em=em-1
1260 IF em<1 THEN GO TO 4500

1300 BEEP .03,45: LET se=se-ez
1310 IF ez+we+INT (RND*3)>VAL a$
+sh+ar+(wa=1) THEN LET am=am-1
1320 IF am<1 THEN GO TO 4000
1330 IF sa>am THEN LET sa=am
1340 IF se>em THEN LET se=em

1400 PRINT AT 20,0;sa;" ";AT 20,
20;se;" ": IF ex-x<3 THEN GO TO
1100
1410 FOR f=1 TO 2: IF sa<am THEN
 LET sa=sa+1: NEXT f
1420 IF se<em THEN LET se=se+1
1430 GO TO 1100

4000 PRINT INK 0;AT y,x;"    ";v$
;"A "; INK 7;AT y+1,x;" CD"; INK
 0;" E "
4010 PRINT AT 20,0;"You die.": S
TOP

4500 PRINT INK 7;AT y,x;" B";w$
;"  ";AT y+1,x;" F "; INK 0;"CD
```

```
4510 PRINT AT 20,20;"He dies.":
STOP

7000 PRINT "Knight Fight"''

7050 RANDOMIZE : LET ah=1: LET e
h=1: LET x=26: LET y=10
7060 LET cash=100: LET sh=0: LET
 ac=0: LET ec=7
7100 PRINT '"(1) Chain-mail cost
s 40 coins,"'"(2) plate-mail 50.
"
7110 PRINT '"(1) Light  lances c
ost 10"'"(2) medium lances 20"'"
(3) heavy  lances 30."
7120 PRINT '"(1) Swords cost 20"
'"(2) axes 20"'"(3) maces 30."''
Shields 20."
7130 FOR f=1 TO 21: FOR g=0 TO 7
: READ a: POKE USR CHR$ (f+143)+
g,a: NEXT g: NEXT f

7200 PRINT AT 21,0;"What armour
do you buy?": LET a$=INKEY$: IF
a$<"1" OR a$>"2" THEN GO TO 7200
7210 BEEP .5,0: LET ar=VAL a$: L
ET cash=cash-30-10*ar
7220 PRINT AT 21,0;"What lance d
o you buy? ": LET a$=INKEY$: IF
a$<"1" OR a$>"3" THEN GO TO 7220
7230 BEEP .5,0: LET la=VAL a$: L
ET cash=cash-10*la
7240 PRINT AT 21,0;"What other w
eapon do you buy?": LET a$=INKEY
$: IF a$<"1" OR a$>"3" THEN GO T
O 7240
7250 BEEP .5,0: LET wa=VAL a$: L
ET w$=CHR$ (152+wa): IF cash-20-
10*(wa=3)<0 THEN GO TO 7240
7260 LET cash=cash-10-10*(wa=3):
 IF cash<20 THEN GO TO 7300
7270 PRINT AT 21,0;"Do you buy a
 shield? (y/n)   ": LET a$=INKEY$
: IF a$<>"y" AND a$<>"n" THEN GO
 TO 7270
7280 BEEP .5,0: LET cash=cash-20
*(a$="y"): LET sh=(a$="y")

7320 LET sa=25: LET se=20: LET a
m=25: LET em=20
7360 CLS : PRINT ''"Your opponen
t chooses:-"''
7370 GO TO 7400+INT (RND*2)*50
```

```
7400 PRINT "Defence: Medium armo
ur."''"Attack: Heavy lance and a
 mace."
7410 LET er=2: LET es=0: LET we=
3: LET le=3: LET v$="J": GO TO 8
000

7450 PRINT "Defence: Heavy armou
r and shield"''"Attack: Medium l
ance and sword."
7460 LET er=2: LET es=1: LET we=
1: LET le=2: LET v$="G"

8000 PRINT ''"(Press ENTER to st
art)": PAUSE 0: CLS
8010 PLOT 0,120: DRAW 250,0: PLO
T 0,120: DRAW 0,30: GO SUB 8500
8020 DRAW 0,-20: GO SUB 8500: DR
AW 0,30: GO SUB 8500
8030 DRAW 0,-10: GO SUB 8500: DR
AW 0,10: GO SUB 8500
8040 DRAW 0,-30: GO SUB 8500: DR
AW 0,20: GO SUB 8500: DRAW 0,-30
8050 PLOT 115,120: DRAW 0,10: DR
AW 20,0,-PI: DRAW 0,-10
8060 FOR f=15 TO 230 STEP 70: PL
OT f,130: DRAW 5,0: DRAW 0,7: DR
AW -5,0,PI: DRAW 0,-7: NEXT f: G
O TO 9000

8500 FOR f=1 TO 3: DRAW 0,5: DRA
W 5,0: DRAW 0,-5: DRAW 5,0: NEXT
 f: DRAW 0,5: DRAW 5,0: DRAW 0,-
5: RETURN

9000 DATA 1,10,15,23,31,59,51,33
,238,61,191,255,255,255,255,255,
0,128,224,240,248,244,243,224
9010 DATA 3,6,6,6,4,3,0,0,255,22
7,128,0,0,0,0,0,192,192,112,56,8
,4,2,6
9020 DATA 1,1,0,1,3,2,6,0,254,24
8,192,128,1,2,2,0,224,192,64,128
,0,0,0,0
9030 DATA 128,60,240,232,246,220
,204,132,55,168,253,255,255,255,
255,255
9040 DATA 0,1,7,15,31,47,207,7,1
28,128,0,128,192,64,96,0,127,31,
3,1,128,64,64,0,7,3,2,1,0,0,0,0
9050 DATA 192,96,16,16,32,192,0,
0,255,199,1,0,0,0,0,0,3,3,14,28,
16,32,64,96
```

```
9060 DATA 120,248,112,56,40,246,
214,116,30,31,14,28,20,111,107,4
8,0,0,0,0,0,0,255,255

9100 DATA 0,0,32,63,0,71,120,0,3
,4,9,251,255,248,8,240
9110 DATA 60,60,60,60,54,34,34,1
02,60,60,60,60,108,68,68,54
9120 DATA 0,64,32,16,10,4,10,3,0
,0,48,112,72,8,4,3,16,24,60,24,1
6,8,4,3
9130 DATA 0,2,4,8,60,32,80,192,0
,0,12,14,18,16,32,192,8,24,60,24
,16,32,64,192
9140 RETURN
```


8   He dies.

Space Raider

To play

In this Spectrum game you control a small spaceship using the 5, 6, 7 and 8 keys. You are under continuous attack from enemy spaceships which, true to the galactic code of honour, attack one at a time.

Your objective is to get as high a score as you can before getting destroyed.

You fire by pressing the space key. Your range of fire includes the 4 character squares directly in front of your ship. Enemy ships destroy you by landing on top of your ship. There are three types of enemy, shown below.
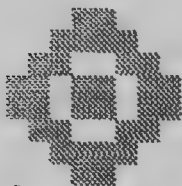
Enemy Globeship
10 Points
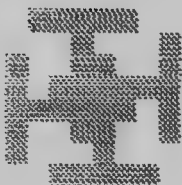
Flickership
20 Points
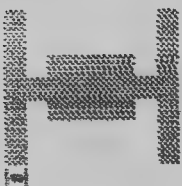
Reflectorship
25 Points
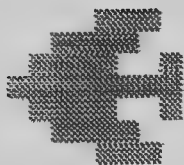
Your Spacecraft

The globeships are the easiest to kill. These are bright green and are worth 10 points each. They sometimes simply abandon ship, leaving an empty, unmoving hulk – you don't get points for destroying the hulk.

The flickerships are much more difficult to destroy, and are worth 20 points each. They constantly change colour, and may be dark and difficult to see, or even temporarily invisible!

The reflectorships are worst of all. To confuse its prey, a reflectorship can produce an image of itself attacking from the opposite direction. You are thus confused by two ships, one real and one fake. Reflectorships are white, and worth 25 points each.

score=0
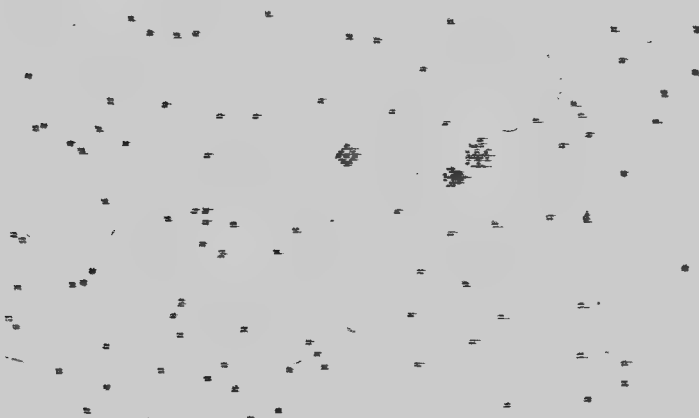
You may initially find it difficult to score at all and you can increase your chances at first by just fighting the green globe-ships. You do this by changing line 330, which chooses one of the three types of enemy. Just change (RND∗3) to (RND∗1).

When you have gained experience in manoeuvring and accurate firing, go back to (RND∗3).

A highscore feature is included.

Notes on listing

10 rs is the current highscore.

20–30 The setting is a black area of space dotted with stars.

50–60 nm is set to 1 if there is an enemy on screen.
ax gives the ship direction, sc is the score, q and r are the co-ords of the enemy ship.

100 This gives the user-defined characters for your spaceship pointing in four directions, a missile and an explosion.

110 x and y are co-ords of your ship.

200 Print your ship pointing in right direction.

210 If enemy on screen goto correct enemy routine.

220 If no enemy, 10 per cent chance one will appear, otherwise goto your move.

300–350 Select enemy and put him at edge of screen at random. sca is the score for killing that enemy, and fgt the line number of the enemy move and attack routine.

360 GOSUB enemy move and attack routine.

400–470 You press button. If you move, goto 500, if you fire then print missile, make firing sound, move missile and see if enemy hit. If he is, goto 2000. Otherwise go back to 210 and repeat loop.

500 You move. X1 and Y1 give direction of movement. Check if enemy on top of new position, and destroys you at line 1000. Set ax to direction of spaceship.

600 Print red spaceship at new position and redo main loop.

1000 You are destroyed. Warning beep given.

1010–1080 Score given, highscore shown and new game offered. Note that if refused, ink, paper and border are reset to normal colours.

2000 Enemy killed. Print exploding enemy ship, give short

triumphant trill, and increment score. Set nm to 0 (no enemy on screen) and enemy co-ords at 0.

3000–3050 Globeship. 3010 gives the move for all ships. A function could have been defined to do this but was found to be slower. Random factor in movement gives unpredictable jumping of enemy ship and makes prediction of enemy's next position difficult.

3020 Draw green globeship.

3030 If ship gets you, goto 1000.

3040 5 per cent chance they abandon ship, leaving hulk.

3100–3140 Flickership. 3120 gives random colour including black – invisible.

3200–3250 Reflectorship.

3220–30 Print white ship and image moving from opposite side of screen. If you are near a corner of the screen the image will move away from you, but if you are near the centre of the screen, you will be confused, often, for the vital seconds until the real ship attacks . . .

8000–8080 DATA for user-defined graphics.

8000–8030 Your ship, pointing in all four compass directions.

8040 Missile.

8050 Explosion.

8060–8080 Enemy ships. These also contain points for destroying that ship and line number where that ship starts its attack routine.

Listing

```
   1 REM Space Raider
  10 LET rs=0: RANDOMIZE
  20 BRIGHT 1: PAPER 0: BORDER 0
: INK 6: CLS
  30 FOR f=1 TO 100: PLOT RND*25
0,RND*150: NEXT f
  40 RESTORE : PRINT AT 0,0;"sco
re=0"
  50 LET NM=0: LET ax=5. LET sc=
0
  60 LET q=0: LET r=0
```

```
 100 FOR f=1 TO 6: FOR g=0 TO 7:
 READ a: POKE USR CHR$ (143+f)+g
,a: NEXT g: NEXT f
 110 LET x=10: LET y=10

 200 INK 2: PRINT AT y,x;CHR$ (1
39+ax)
 210 IF nm THEN GO TO 360
 220 IF RND<.9 THEN GO TO 400
 300 LET nm=1: LET q=1
 310 IF RND<.5 THEN LET q=30
 320 LET r=INT (RND*20)
 330 LET ns=INT (RND*5)*10: REST
ORE 8050+ns
 340 FOR f=0 TO 7: READ a: POKE
USR CHR$ 150+f,a: NEXT f
 350 READ sca: READ fgt

 360 GO SUB fgt

 400 LET ix=CODE INKEY$: IF ix>5
2 AND ix<57 THEN GO TO 500
 410 IF ix<>32 THEN GO TO 210
 420 LET bx=(ax=8)-(ax=5): LET b
y=(ax=6)-(ax=7)
 430 INK 6: FOR f=1 TO 5: LET zx
=x+f*SGN bx: LET zy=y+f*SGN by
 440 IF zx<1 OR zx>30 OR zy<1 OR
zy>20 THEN GO TO 210
 450 PRINT AT zy,zx;CHR$ 148;AT
zy,zx;" ": IF zx=q AND zy=r THEN
 GO TO 2000
 460 BEEP .01,1: BEEP .01,10: NE
XT f
 470 GO TO 210
 500 LET ax=ix-48
 510 LET x1=(ax=8)-(ax=5)+(x<1)-
(x>30): LET y1=(ax=6)-(ax=7)+(y<
1)-(y>20)
 520 LET x=x+x1: LET y=y+y1
 530 IF SCREEN$ (y,x)=CHR$ 150 T
HEN GO TO 1000

 600 INK 2: PRINT AT y-y1,x-x1;"
 ";AT y,x;CHR$ (139+ax)
 610 GO TO 210

 1000 FOR f=1 TO 30: BEEP .05,20:
 BEEP .05,4: NEXT f
 1010 INK 7: PRINT AT 5,2;"Your f
inal score is ";sc
 1020 IF sc>rs THEN LET rs=sc
```

```
1030 PRINT : PRINT "Best so far
is ";rs
1040 PRINT : PRINT "another game
?"
1050 IF INKEY$="y" THEN GO TO 20
1060 IF INKEY$="n" THEN GO TO 10
80
1070 GO TO 1050
1080 INK 0: PAPER 7: BORDER 7: S
TOP

2000 FOR f=1 TO 10: PRINT AT r,q
;CHR$ 149: BEEP .1,20: PRINT AT
r,q;" ": BEEP .1,-4: NEXT f
2010 FOR f=1 TO 50: BEEP .05,60-
f: NEXT f
2020 LET nm=0: LET sc=sc+sca
2030 INK 7: PRINT AT 0,0;"score=
";sc
2040 LET r=0: LET q=0
2050 LET x1=0: LET y1=0: GO TO 6
20
3000 PRINT AT r,q;" "
3010 LET q=q+(q<x)-(q>x)+INT (RN
D*3-1): LET r=r+(r<y)-(r>y)+INT
(RND*3-1)
3020 INK 4: PRINT AT r,q;CHR$ 15
0
3030 IF q=x AND r=y THEN GO TO 1
000
3040 IF RND<.05 THEN LET nm=0
3050 RETURN
3100 PRINT AT r,q;" "
3110 LET q=q+(q<x)-(q>x)+INT (RN
D*3-1): LET r=r+(r<y)-(r>y)+INT
(RND*3-1)
3120 INK RND*7: PRINT AT r,q;CHR
$ 150
3130 IF q=x AND r=y THEN GO TO 1
000
3140 RETURN

3200 PRINT AT r,q;":";AT 20-r,30
-q;"."
3210 LET q=q+(q<x)-(q>x)+INT (RN
D*3-1): LET r=r+(r<y)-(r>y)+INT
(RND*3-1)
3220 INK 7: PRINT AT r,q;CHR$ 15
0
3230 PRINT AT 20-r,30-q;CHR$ 150
3240 IF q=x AND r=y THEN GO TO 1
000
3250 RETURN
```

```
8000 DATA 14,60,121,255,121,60,1
4,0
8010 DATA 56,146,214,254,124,124
,56,16
8020 DATA 16,56,124,124,254,214,
146,56
8030 DATA 112,60,158,255,158,60,
112,0
8040 DATA 0,0,8,56,28,8,0,0
8050 DATA 65,8,0,146,0,65,4,0
8060 DATA 24,60,102,219,219,102,
60,24,10,3000
8070 DATA 124,17,153,191,253,153
,136,82,20,3100
8080 DATA 0,129,129,189,255,189,
129,129,25,3200
```

Probability Check

This Spectrum program simulates a ball dropping on to a triangle of pegs. On hitting a peg, the ball bounces right or left at random, before falling to the next layer. This is repeated until the balls are counted at the bottom of the screen.

This gives an idea of how accurate probability theory is for small numbers of events. You have control over two factors: the number of runs (events) and the number of choices of routes. The number of routes must be between 2 and 8, but there is no maximum limit to the number of runs.

For example, when tossing three coins, there are four possible results: 3 heads, 3 tails, 2 heads and 1 tail, or 2 tails and 1 head. However, there is only one way of getting 3 heads or 3 tails, but three ways each of getting 2 heads and 1 tail or 2 tails and 1 head. Thus in 200 throws, you would expect to get $3/8 \times 200 = 75$ results of 2 heads $+ 1$ tail, and $1/8 \times 200 = 25$ results of three heads. We can simulate this in the program. Enter 200 for 'number of runs' and 4 for number of choices. See how far your results agree with the predicted ones, the choices being right and left instead of heads and tails.

Notes on listing
120 NU is number of events. Note this is corrected to a positive integer if necessary.
140 NC = number of choices, corrected to between 2 and 8.
150 The final 'field' is yellow, with a green border, blue 'pegs' and a red 'ball'.
200–240 Print triangle of pegs, with straight channel after bottom layer, one for each choice.
300–360 Each ball starts at print position 1, 16.

```
Size of sample=100
Choices=3
                              .        run no.100


                         .         .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .
                  .      .      .      .

                  32     43     25



Size of sample=1000
Choices=8
                              .        run no.201

                         .    o    .
                     .        .        .
                 .        .        .        .
             .        .        .        .        .
         .        .       .        .      .      .
     .       .        .        .        .        .
 .       .        .        .        .        .        .      .
 .       .        .        .        .        .        .      .    .
 .       .        .        .        .        .        .      .    .

     2      10     28     59     58     29     13     1
```

340 checks if peg underneath ball,
  350 if ball has finished falling.
400–410 Ball bounces randomly left or right on hitting peg.
500–520 Print number of balls in channel so far. Note that this

is done by checking the X co-ordinate of the ball when it lands.

Repeat for NU runs.

Listing

```
   1 REM Probability check
   2 REM  © S.Robert Speel 1982

 100 PRINT "Probability check"
 110 PRINT : PRINT "How many run
s?"
 120 INPUT nu: LET nu=INT ABS nu
 130 PRINT "how many choices? (2
 TO 8)"
 140 INPUT nc: LET nc=INT ABS nc
: LET nc=nc+(nc<2)*(2-nc)-(nc>8)
*(nc-8)
 150 PAPER 6: BORDER 4: CLS
 160 PRINT "Size of sample=";nu:
 PRINT "Choices=";nc;AT 2,20;"ru
n no. "

 200 INK 1: FOR f=1 TO nc-1: FOR
 g=-f TO f-2 STEP 2
 210 PRINT AT f*2,18+g*2;"."
 220 NEXT g: NEXT f
 230 PRINT : FOR f=nc*2 TO 18: F
OR g=-nc-1 TO nc STEP 2
 240 PRINT TAB 18+g*2-(g=7);".";
: NEXT g: NEXT f
 250 DIM a(8)

 300 INK 2: FOR f=1 TO nu
 310 LET x=16: LET y=1
 320 PRINT AT 2,27;f
 330 PRINT AT y,x;"o"
 340 IF SCREEN$ (y+1,x)="." THEN
 GO TO 400
 350 IF y>18 THEN GO TO 500
 360 LET y=y+1: PRINT AT y-1,x;"
 "; GO TO 330

 400 PRINT AT y,x;" "
 410 LET x=x+INT (RND*3-1)*2: GO
 TO 320

 500 PRINT AT y,x;" ": LET ch=(1
+x)/4
 510 LET a(ch)=a(ch)+1
 520 PRINT AT 20,x;a(ch): NEXT f
```

Wave Addition

This is a short Spectrum program designed to show constructive and destructive interference between waves. It uses polychrome high-resolution graphics very effectively.

To use
When RUNning the program, all you decide is the phase change between the two waves; that is, how far apart the peaks will be. If the waves are on top of each other, then they are in phase. If the peaks are separate, the waves are out of phase.

After you enter the phase change, the computer PLOTs a graph showing the two waves, and their sum. If the sum is far from the horizontal axis, there is constructive interference. When the sum is near or on the horizontal axis, the waves cancel each other out – this is destructive interference.

The maximum sum of the waves is also given.

Note that if waves are completely out of phase, dotted line lies on horizontal axis.

Notes on listing
10 Graphic A is used for the PI symbol.
20 INPUT distance between peaks of the two waves.
40 DRAW horizontal lines and axis.
50 DRAW vertical axis.
60 Scale on vertical axis.
70 PRINT phase change.
80 Horizontal scale using *graphic* A.
90 X4 used to calculate maximum sum of waves.
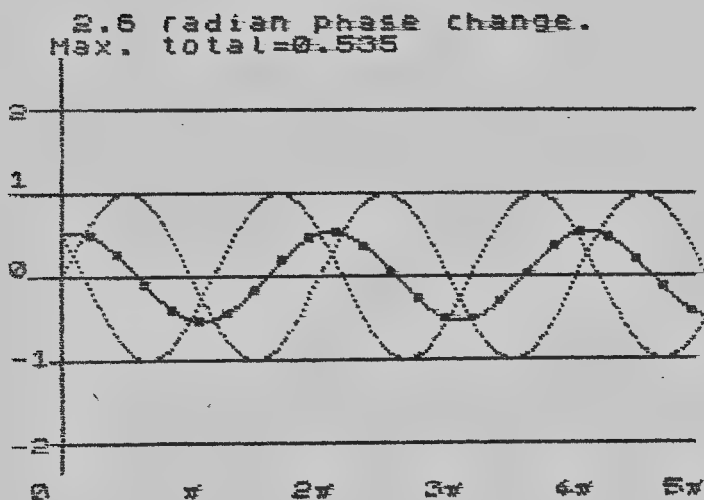
0 radian phase change.
Max. total=2
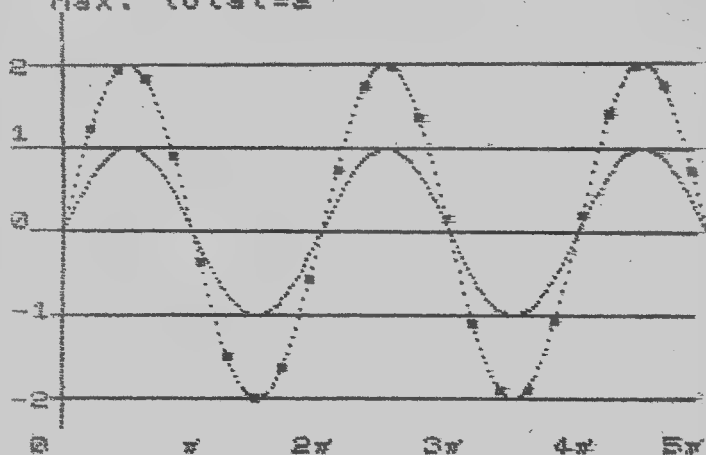
Waves in phase

PI/2 radian phase change.
Max. total=1.414
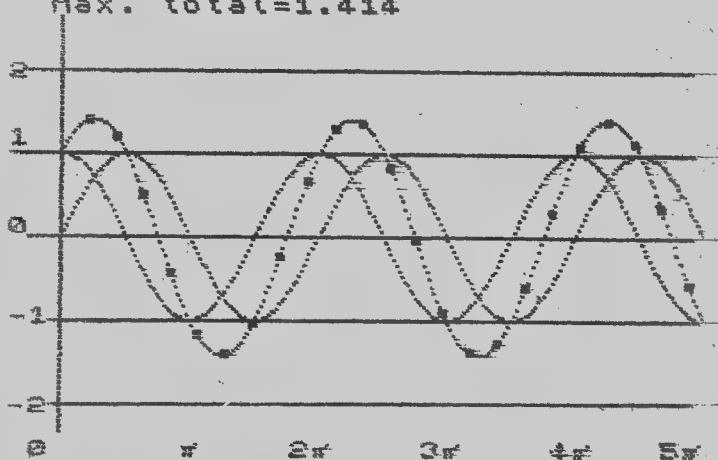
Waves slightly out of phase

2.6 radian phase change.
Max. total=0.535

Waves almost completely out of phase

100–160 PLOT 3 waves. 2 waves in red and blue, sum in black. X1 and X2 are points on 2 waves, X3 is point on sum of waves.

150 Every tenth wave goto PLOT heavy dot routine.

170 Calculate maximum point of wave using X4.

200–220 PLOT heavy dot routine.

8000 DATA for Greek letter PI.

Listing

```
      1 REM Wave Addition

    10 FOR f=0 TO 7: READ a: POKE
USR "A"+f,a: NEXT f
    20 PRINT "input phase change":
 INPUT p$: LET ph=VAL p$
    30 INK 0: PAPER 6: BORDER 3: C
LS
    40 FOR f=0 TO 4: PLOT 1,20+30*
f: DRAW 250,0: NEXT f
```

```
   50 PLOT 20,10: DRAW 0,150
   60 PRINT AT 4,0;"2";AT 7,0;"1"
;AT 11,0;"0";AT 15,0;"-1";AT 19,
0;"-2"
   70 PRINT AT 0,3;p$;" radian ph
ase change."
   80 PRINT AT 21,1;"0          A
2A   3A     4A    5A"
   90 LET x4=0

  100 FOR f=1 TO PI*75
  110 LET x1=80+SIN (f/15)*30
  120 LET x2=80+SIN (f/15+ph)*30
  130 LET x3=x2+x1-80: IF x3>x4 T
HEN LET x4=x3
  140: INK 1: PLOT f+20,x1: INK 2
: PLOT f+20,x2: INK 0: PLOT f+20
,x3
  150 IF f/10=INT (f/10) THEN GO
TO 200
  180 NEXT f

  170 PRINT AT 1,2;"Max. total=";
INT ((x4-80)/3*100+.5)/1000

  200 FOR g=-1 TO 1: FOR h=-1 TO
1
  210 PLOT f+20++g,x3+h: NEXT h:
NEXT g
  220 GO TO 160

8000 DATA 0,0,2,124,40,40,40,0
```

Converting ZX81 Programs for Use on the Spectrum

Although ZX81 BASIC is essentially a sub-set of ZX Spectrum BASIC, there are several significant alterations which have to be made when converting ZX81 programs for use on the Spectrum.

For our purposes we can divide programs into three categories:

(a) Programs consisting mainly of words or calculations. The mathematics, PRINTing and string commands have altered little, and many Adventure-type games, mini word processor, mathematical and quiz-type programs can be virtually copied from the ZX81 listing with only a few modifications. (These types are also easy to convert from Spectrum to ZX81, bearing in mind memory limitations.)

(b) Programs which are mainly words or calculations but with supporting diagrams. This includes many scientific and maths programs where a picture helps to explain the calculations, as well as a lot of educational programs using a 'reward' (such as a drawing of a pattern or animated picture if you successfully complete a test). Here there is some difficulty in conversions, and often it is simpler to devise a new routine on the Spectrum to replace the old graphic routine.

(c) Programs highly dependent on moving graphics or pictures. This covers a whole range of games as well as serious programs to do with map-making, drawing, planning, designing and some maths programs. To convert these, you usually need a very clear idea of exactly what the program does and what it shows on the screen.

Here is a list of conversions which you may need to make when translating to Spectrum BASIC.

PAUSE

To pause until a button is pressed on the ZX81, PAUSE 40000 (or any number larger than 32768) is used. This should be changed to PAUSE 0 on the Spectrum. The screen gives a flicker when PAUSE is used on the ZX81, and often PAUSE is avoided and an empty FOR-NEXT loop used instead. This may sometimes be replaced by PAUSE on the Spectrum.

SLOW *and* FAST

The ZX Spectrum operates at the speed of the ZX81 in FAST mode with the steady display of SLOW mode. For games using animated graphics, you may need to slow down the program on the Spectrum. Do not use PAUSE, because this will not work when a key is pressed; use instead an empty FOR-NEXT loop, e.g.,

FOR F = 1 TO 20: NEXT F

You could also slow down by adding BEEP statements, giving you sound as a bonus.

SCROLL

The Spectrum scrolls automatically, which is all very well, but the SCROLL key command has been omitted. This is a nuisance in games which rely on scrolling when the screen is not full, such as Alien Descender and Asteroid Belt. Also in word-based games, if the word 'scroll?' appears at the bottom of the screen, an inexperienced user might press the largest key in sight – the space key – which will BREAK the program.

Fortunately, there is a POKE we can do which allows for scrolling without the 'scroll?' at the bottom of the screen. Then we need to PRINT too much on screen, so that we force the scroll. Therefore replace each SCROLL in a ZX81 program with

POKE 23692, 255: PRINT AT 21,0: PRINT

PEEKing and POKEing on screen
In many ZX81 programs you will see commands such as

```
PRINT AT Y,X;
LET  A$ = CHR$ PEEK (PEEK  16398+256*PEEK
16399).
```

Addresses 16398 and 16399 hold the address of the screen
position last PRINTed at, so PEEKing this value gives the code
of the character at that point (in this example, position Y,X).
Conversely, POKE (PEEK 16398+256*PEEK 16399), n
POKEs the character with code n into the screen.

On the Spectrum, the PEEK can be replaced with

```
LET A$ = SCREEN$ (Y,X)
```

which gives the character at Y,X, or

```
LET A = ATTR(Y,X)
```

which gives the ATTRibutes of the character – colours, bright-
ness and whether it is flashing. To replace the POKE is
awkward, as the characters on screen are stored in an odd way
in the Spectrum. You will have to use PRINT AT instead of
a POKE command.

Timing
On the ZX81 timing routines may be made by altering a
variable every time a loop is executed. This may be slowed to
a convenient value (e.g., one loop per second) with a PAUSE,
or a dummy command such as LET A = RND. Alternatively,
a systems variable called FRAMES (PEEK 16436+256*
PEEK 16437) may be PEEKed at intervals. This is used with
PAUSE, and can accurately keep time for up to about eleven
minutes before needing to be reset.
On the Spectrum, these timing devices can be replaced by
the Spectrum FRAMES, which, as it is a 3-byte counter

(PEEK 23672+256*PEEK 23673+65536*PEEK 23674), will give the time in fiftieths of a second for nearly four days before running down. Remember to POKE these values to zero when starting the timer. Also, PEEK these values twice and take the larger value to get the time (see manual for reasons).

Multi-line statements

It is a good idea to 'compact' ZX81 programs by putting several variables in one multi-line statement rather than using one variable per program line. Small FOR-NEXT loops, dimensioned strings and other routines can be conveniently reduced to fewer lines.

PLOT and UNPLOT

On the ZX81, PLOT uses pixels which are one quarter of the size of a character square. This gives 44 rows and 64 columns of PLOT co-ordinates. On the Spectrum, the PLOT command uses high-resolution – 256 points by 176. This means that fine lines, circles and pictures can be made which are impossible on the ZX81. Problems arise when converting the large ZX81 plotted pixels to the finer Spectrum points.

If the ZX81 is using a routine to draw lines or pictures, you may be able to replace these with the DRAW and CIRCLE commands. Where the ZX81 PLOTs a more solid shape, the Spectrum would give an outline only. Try replacing the ZX81 routine with a PRINT AT routine.

By using PLOT OVER 1, unplot can be achieved on the Spectrum.

Memory

The Spectrum has colour and high-resolution graphics. This means that on a 16K machine nearly 7K is reserved for the screen, and you can only use 9K. So a 16K ZX81 program may be difficult to fit on to a 16K Spectrum. To cut down a ZX81 program, use multi-line statements. Also, look out for common routines which could be replaced by user-defined functions, and IF – THEN routines which you can compact, e.g.,

```
100 IF A$ = "YES" THEN GOSUB 700
    .
    .
    .
    .
    .
    .
700 PRINT 'YOUR SCORE IS 0'
710 LET K = 14
720 LET Y = Y − 1
730 RETURN
```

can, on the Spectrum, be compacted to

```
100 IF a$ = 'yes' THEN PRINT 'Your score is 0': LET K
= 14: LET y = y − 1
```

Adding sound, colour and user-defined graphics
Naturally you will want to make full use of your ZX Spectrum's superior features. However, don't add sound, colour and UDGs until you have the converted program working. If you add as you convert, there is a greater chance that the program will end up tortuous and convoluted.

Patterns

With some quite short programs the ZX81 can draw very interesting patterns. Not only do they look good at any stage, as on a print-out, but also they are interesting to watch. The pattern is endlessly changing, growing and reforming.

These patterns are fascinating to look at, but they have other uses too. For instance, they can be incorporated into other programs as rewards. When you complete a Fairway round in ten shots, instead of just a message such as YOUR TOTAL NUMBER OF SHOTS WAS 10, a pattern would appear to reward you. This greatly enhances many games and serious programs too.

This sort of program often appeals greatly to people who are not generally interested in computers. The pattern programs show off the ZX81's graphic capabilities to a great extent and may help to condition a non-computer-loving family to the fact that you must have a printer, more cassettes, etc.

Pattern 1 – Diagonals

Listing occupies 0.4K
Program runs in 1.3K

This program draws a continuously changing pattern by moving a ball around the screen which leaves a trail of graphic characters. Every now and then the graphic character changes, and a pattern builds up. Only graphic characters between codes 128 and 138 are used.
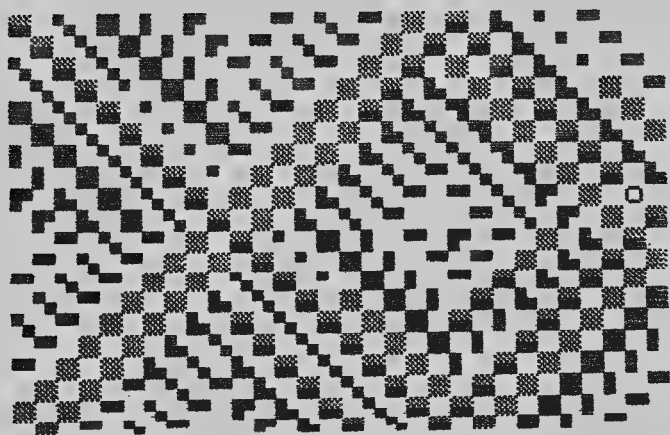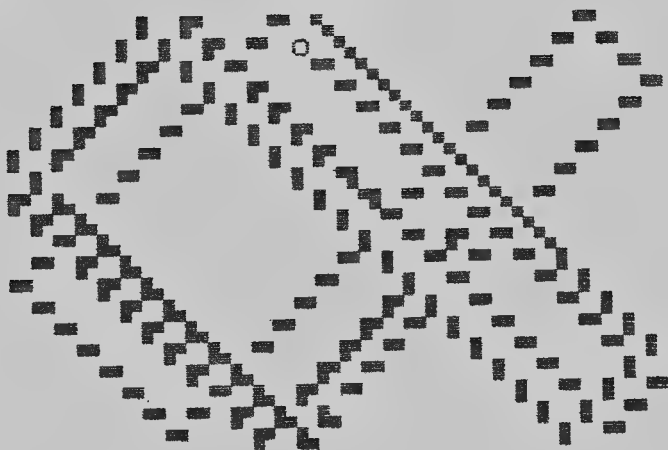
Notes on listing

50–110 X + Y are co-ords of ball.

A$ is the character to be printed when ball moves. X1 + Y1 are the speeds north and east. (When negative they are south and west.) Note that the ball can start in any diagonal

direction but cannot go straight north, south, east or west since this would result in a continual bouncing off in one line.

500 Print ball, so that you know what the pattern is doing at present. This line can be removed if you dislike the ball.

510–520 If ball 'hits' side of screen, make it bounce. Note that a border of one square is left all around.

600–610 Change X+Y according to X1+Y1.

620 Print current graphic where ball was.

630 Change graphic randomly to next graphic. RND <.05 may seem a very little chance of changing a graphic, but in ten moves, the chance of changing at least once is $1-(0.95)^{10}$, which is 0.4. So the graphics do change quite often.

640 Repeat loop.

Listing

```
   1 REM PATTERNS 1
   2 REM      COPYRIGHT
     S.ROBERT SPEEL 1982
  10 RAND
  50 LET X=10
  60 LET Y=10
  70 LET A$="▟"
 100 LET X1=1-2*(RND<.5)
 110 LET Y1=1-2*(RND<.5)
 500 PRINT AT Y,X;"O"
 510 LET X1=X1-2*(X=30)+2*(X=1)
 520 LET Y1=Y1-2*(Y=20)+2*(Y=1)
 600 LET X=X+X1
 610 LET Y=Y+Y1
 620 PRINT AT Y-Y1,X-X1;A$
 630 IF RND<.05 THEN LET A$=CHR$
     ((CODE A$+1)-11*(A$="▓"))
 640 GOTO 500
```
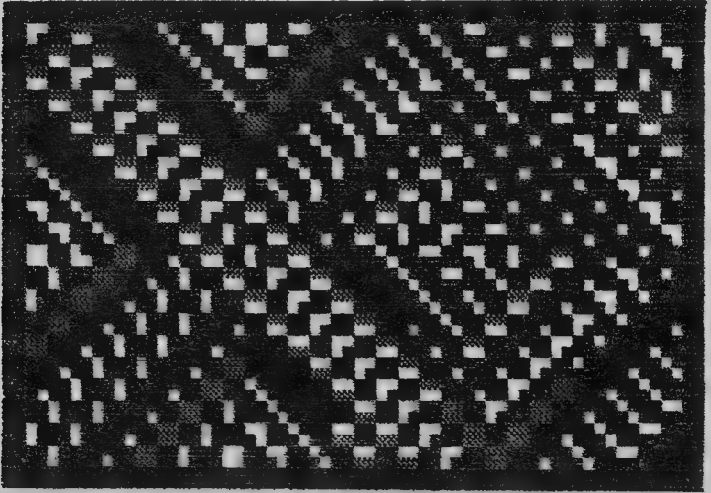
Pattern 2 – Reversal Diagonals

Listing occupies 0.5K
Program runs in 1.5K

This is a development of Pattern 1.

Instead of using just graphics with codes 128–138, it uses other graphics, codes 0–10, as well.

Also, the picture is drawn on a black background rather than on the white screen. This looks less effective printed out on paper but is very striking on screen.

There are many ways in which you can alter this program. If you wish a more frequent change of graphics, change the value of RND in line 630 to RND <.1 or even larger. You could even omit the RND part, so the graphic changes every go. You could also change the starting position of the ball, leading to a more (or less) symmetrical pattern. The bounce angle could be made random, producing a less symmetrical pattern.

Notes on listing
The only changes to the listing of Pattern 1 are:
20–40 Draw a black background. Note that a black border will always be left around the pattern.
80 The pattern starts with a white square as the initial graphic.
630 The limits on graphics have been changed so that graphics 0–10 are included. Note that if the line read:

```
630 IF RND<.05 THEN LET A$=CHR$
((CODE A$+1)+118*(A$="▒")-138*(
A$="▓"))
```

then black and white squares would not be included in the
possible graphics.

Listing

```
  1 REM PATTERNS 2
 10 REM     COPYRIGHT 1982
            S.ROBERT SPEEL
 20 FOR F=1 TO 22
 30 PRINT "
           "
 40 NEXT F
 60 LET X=10
 70 LET Y=10
 80 LET A$="  "
100 LET X1=1
110 LET Y1=1
500 PRINT AT Y,X;"0"
510 LET X1=X1-2*(X=30)+2*(X=1)
520 LET Y1=Y1-2*(Y=20)+2*(Y=1)
600 LET X=X+X1
610 LET Y=Y+Y1
620 PRINT AT Y-Y1,X-X1;A$
630 IF RND<.05 THEN LET A$=CHR$
((CODE A$+1)+117*(A$="▒")-139*(
A$="▓"))
640 GOTO 500
```

Pattern 2A – Diagonal Zig-zag

One variation of Pattern 2 limits the graphics to just vertical
straight lines.

The background here is made up of horizontal straight lines,
producing a mind-bending pattern such as shown overleaf.

Notes on listing
The changes from Pattern 2 are:
30 Prints horizontal background.
80 A$ is initially a vertical line.
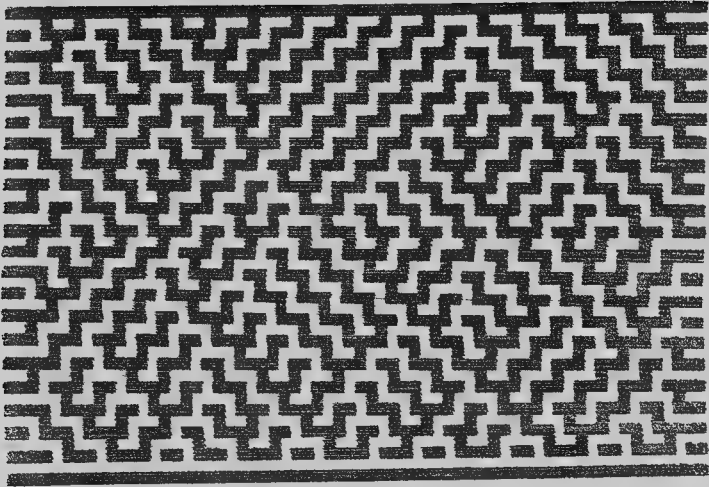
630–640 The only characters used are vertical lines. Note that RND in 640 will produce a change less often than in line 630. This compensates for the possibility of the graphic changing in line 630, and then changing back immediately in line 640.

Listing

```
   1 REM  PATTERNS 2A
  10 REM      COPYRIGHT 1982
             S.ROBERT SPEEL
  20 FOR F=1 TO 22
  30 PRINT "█
             "
  40 NEXT F
  60 LET X=10
  70 LET Y=10
  80 LET A$=" ▌"
 100 LET X1=1
 110 LET Y1=1
 500 PRINT AT Y,X;"O"
 510 LET X1=X1-2*(X=30)+2*(X=1)
 520 LET Y1=Y1-2*(Y=20)+2*(Y=1)
 600 LET X=X+X1
 610 LET Y=Y+Y1
 620 PRINT AT Y-Y1,X-X1;A$
 630 IF RND<.05 THEN LET A$=" ▌"
 640 IF RND<.044 THEN LET A$="▌ "
 650 GOTO 500
```

Pattern 3 – Square Patterns

This program draws a pattern entirely made up of graphic squares. However, due to the continuous forming of new squares, the old ones get drawn over or have a part knocked off, so that there are usually very few whole squares on the screen. The resulting pattern formation is very attractive.

Occasionally squares may be drawn in white, i.e., that area is rubbed out. This stops the screen from becoming too full.

I have made the squares appear in a square format, which leaves the right-hand part of the screen empty. This can be changed to give a full screen format.
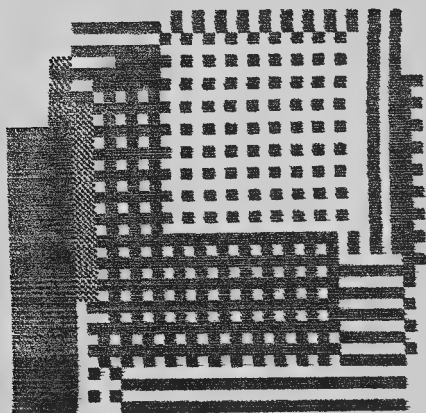
Notes on listing

60 L determines the size of the square.

70–80 M is the code of a graphics character, code 0–10 or 128–138. Change as desired.

100–110 X+Y are the co-ords of the centre of the square. By altering Y to a larger number and changing line 140 from . . . 10+G+Y to, say, 15+G+Y, a full screen format can be achieved.

120–160 Draw square. The +10 in line 140 ensures that the computer does not try to print offscreen.

170 Redo loop. Note that GOTO 50 would be less random, oddly enough. Recursive calls to RAND are not a good idea.

Listing

```
   1 REM  SQUARES PATTERN
  10 REM     COPYRIGHT 1982
               S.ROBERT SPEEL
  50 RAND
  60 LET L=INT (RND*7)
  70 LET M=INT (RND*11)
  80 IF RND<.5 THEN LET M=M+128
 100 LET X=RND*8-4
 110 LET Y=RND*8-4
 120 FOR F=-L TO L
 130 FOR G=-L TO L
 140 PRINT AT 10+F+X,10+G+Y;CHR$
M
 150 NEXT G
 160 NEXT F
 170 GOTO 60
```

Note that sometimes the program appears to stop doing anything for a few moments. This is when it is printing a graphic square on top of a square of the same graphic.
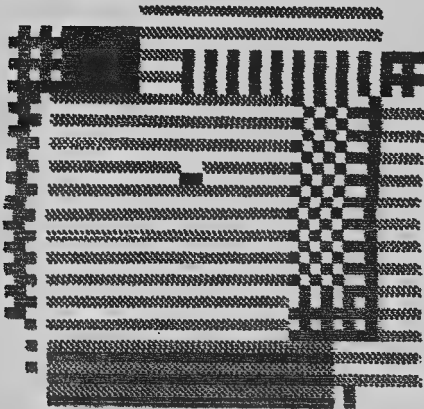
Pattern 4 – Buildup-breakup

This is a more symmetrical pattern than the others. It creates
a graphic pattern randomly, but each time a graphic character
is chosen and positioned, three similar characters are placed
so that the whole pattern is symmetrical, vertically and
horizontally, about the centre of the screen.

Two examples of patterns produced by the program as it
stands are shown, below and overleaf.
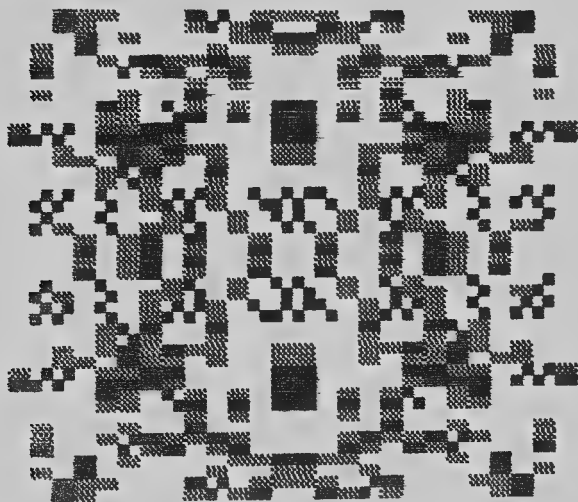
The program can easily be altered to produce other types of pat-
tern by changing A$.

If there are one or more spaces in A$, the pattern will be
open, whereas if there are no spaces, the pattern becomes more
solid.

Notes on listing
50 A$ contains the characters to be used in the pattern. It can
be any length. Try using vertical and horizontal lines only,
or letters.

100–240 Loop for drawing four copies of each character in A$. Note that the loops will occur the number of times you put into A$.

100–120 Position of character, centred about middle of screen.

200–230 Print four identical characters from A$ in square.
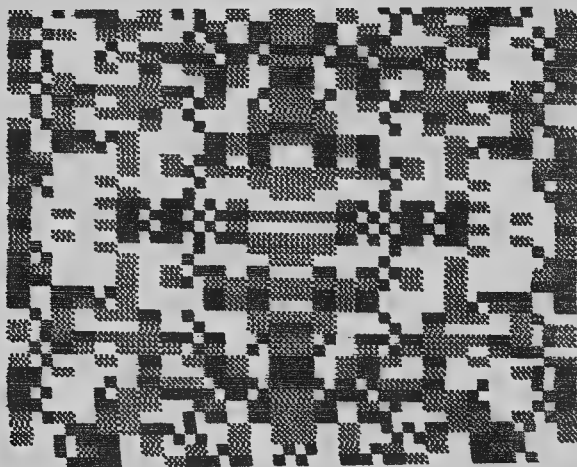
240 Repeat for other characters in A$.

300 Start again at beginning of A$.

Listing

```
  10 REM  BUILDUP-BREAKUP
  20 REM      COPYRIGHT 1982
                S.ROBERT SPEEL
  50 LET A$="          "
 100 FOR F=1 TO LEN A$
 110 LET A=INT (RND*22)
 120 LET B=INT (RND*22)+7
 200 PRINT AT A,B;A$(F)
 210 PRINT AT 21-A,B;A$(F)
 220 PRINT AT A,31-B;A$(F)
 230 PRINT AT 21-A,31-B;A$(F)
 240 NEXT F
 260 GOTO 100
```

Pattern 4A – Buildup-breakup Block

This shows one useful development of Pattern 4.

Instead of printing just one character at a time, we print four. This includes the first character from strings A\$, B\$, C\$ and D\$, printed in a block.

```
A$; B$
C$; D$
```

Again, A\$ can be any length, but B\$, C\$ and D\$ should be the same length. This allows build-up of small pictures and more regular patterns. If A\$ = B\$ = C\$ = D\$, then the block is made of one graphic character repeated four times. This can produce a pattern like this.
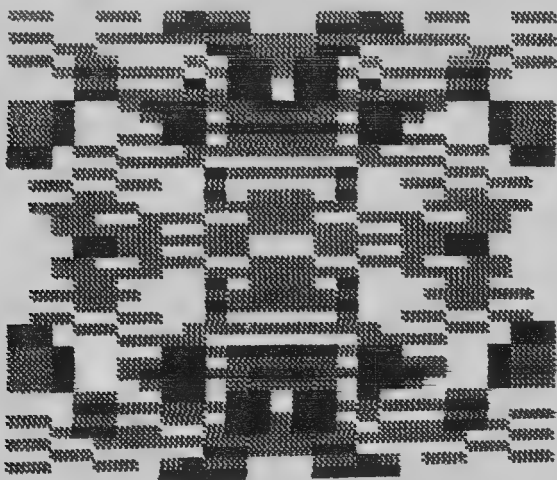
Notes on listing
50–80 A\$, B\$, C\$ and D\$ are the same length. These particular graphics will give a weird 'bubbling' effect on the screen, which looks good in action but not when 'frozen' on paper. Change A\$–D\$ to any values you like, and increase

their length as you wish. Try blocks of characters with codes 6 and 134 only.

110 Note that A's limits have changed; due to printing at A+1, A must not exceed 20.

200–235 Print blocks of characters in usual way with second line on line below first line.

Listing

```
 10 REM  BUILDUP-BREAKUP BLOCK
 20 REM      COPYRIGHT 1982
             S.ROBERT SPEEL
 50 LET A$="█ ▪"
 60 LET B$="█ ▪"
 70 LET C$="█▪"
 80 LET D$="█ ▪"
100 FOR F=1 TO LEN A$
110 LET A=INT (RND*20)+1
120 LET B=INT (RND*22)+7
200 PRINT AT A,B;A$(F);B$(F)
205 PRINT AT A+1,B;C$(F);D$(F)
210 PRINT AT 21-A,B;A$(F);B$(F)
215 PRINT AT 22-A,B;C$(F);D$(F)
220 PRINT AT A,31-B;A$(F);B$(F)
225 PRINT AT A+1,31-B;C$(F);D$(
F)
230 PRINT AT 21-A,31-B;A$(F);B$
(F)
235 PRINT AT 22-A,31-B;C$(F);D$
(F)
240 NEXT F
250 GOTO 100
```

Overlay

Program runs in 1K

This is a 1K pattern which is interesting to watch for quite a long time as it changes and develops from one configuration to another. It makes a good display program, as it repeats itself only at long intervals, each character in line 30 taking fifty seconds to run through. By changing line 30, you can have completely different patterns appearing. You can also have the list

longer than eight graphics and the program will accept this
without further change.
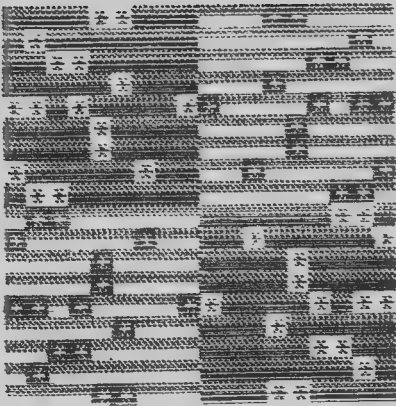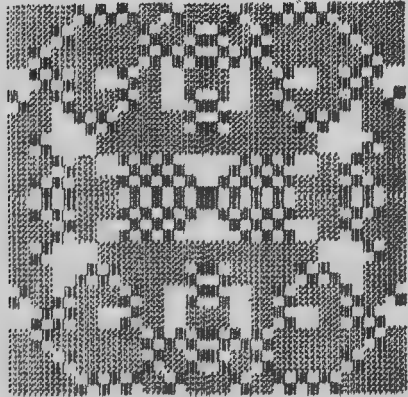
Two patterns produced by the program are shown, without *any*
changes to the listing, even to line 30. This shows how much
just 1K can do for you!

Notes on listing
10 X sets the size of grid.
20 Z acts as a pointer to the character being used in string A$.

30 A$ contains the characters used in the pattern. This can be changed and made longer, to give completely different patterns.

Here is one alteration you could try.

```
30 LET A$="..█.▟▙.▟█X+▓"
```

40–90 Printing loop. Each character is used for about fifty seconds before the next is started.

70 adds 128 to the character's code to get its inverse character, or take 128 off if character is already inverse.

80 PRINT 2 normal and 2 inverse characters in 4 quadrants of picture. By putting a character, immediately followed by its inverse, in line 30, very striking effects can be produced when they change over.

90 Repeat 150 times.

100–120 Set Z to next character in A$ and repeat for new character. If at end of A$, start sequence again.

Listing

```
   1 REM OVERLAY
   2 REM BY S.ROBERT SPEEL
  10 LET X=17
  20 LET Z=1
  30 LET A$="  ▓ O█▄█"
  40 FOR F=1 TO 150
  50 LET B=INT (RND*9)
  60 LET C=INT (RND*9)
  70 LET A=128+CODE A$(Z)-(CODE
A$(Z)>127)*256
  80 PRINT AT B,C;A$(Z);AT B,X-C
;CHR$ A;AT X-B,C;CHR$ A;AT X-B,X
-C;A$(Z)
  90 NEXT F
 100 IF Z=LEN A$ THEN LET Z=0
 110 LET Z=Z+1
 120 GOTO 40
```

Plot Sketch

Program runs in 1K

To use
When you RUN the program, a single pixel appears in the bottom left-hand corner of the screen. You can move this 'cursor' with keys 5, 6, 7 and 8, leaving a black trail. With a little practice, you will be able to draw good pictures. You can move the cursor without leaving a trail by pressing key 0 (RUBOUT). This will make the cursor rub out lines which it passes over. To get back to drawing mode, press key 9 (graphics).

Note that it is possible to run out of memory with very black pictures when using only 1K RAM. With 2 or more K, you won't run out of memory. The pictures above and overleaf were drawn using a 1K machine.

Notes on listing

10–30 X and Y are the co-ords of your cursor. RS is set to 1
   to PLOT, 0 to UNPLOT.

60–70 Move cursor according to key pressed.

80–90 Check that you do not go outside bounds of screen.
   The picture area can be increased if using 2K or more RAM,
   i.e., 80 LET x = x+(x<1)−(x>60)
      LET y= y+(y<1)−(y>40)

100 Change plot/unplot mode if key pressed.
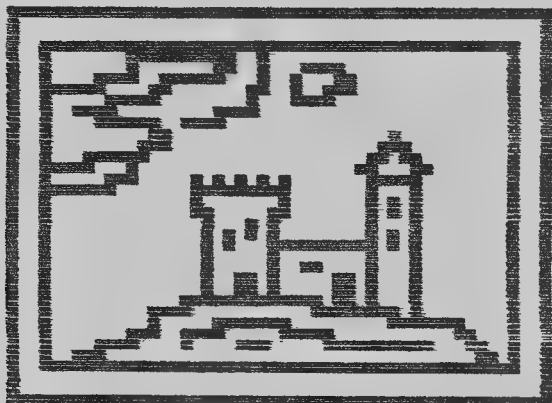
110 Repeat until BREAK pressed.

Listing

```
   1 REM PLOTSKETCH

  10 LET X=1
  20 LET Y=5
  30 LET RS=1

  40 PLOT X,Y
  50 IF RS<1 THEN UNPLOT X,Y
  60 LET X=X+(INKEY$="8")-(INKEY
$="5")
  70 LET Y=Y+(INKEY$="7")-(INKEY
$="6")
  80 LET X=X+(X<1)-(X>50)
  90 LET Y=Y+(Y<5)-(Y>40)

 100 LET RS=SGN (RS+(INKEY$="9")
-(INKEY$="0"))
 110 GOTO 40
```

Screen Art

A program for drawing directly on to the screen.

Listing occupies 0.8K
Program runs in 1.8K

It is possible to make pictures on the screen with the ZX81. This can be used to design garden layouts, graphic designs, drawings with perspective or just for amusement.

On running, a black square appears in the bottom left-hand corner of the screen, flickering. The keys 5, 6, 7 and 8 move the flickering square in the normal directions leaving a black trail. The keys 1, 2, 3 and 4 are also used for moving north-west, north-east, south-east and south-west respectively.
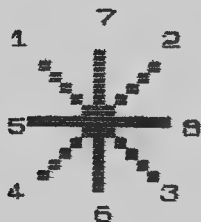
Holding down a key leads to continual moving in that direction. When wishing to move more slowly, one space at a time, press SLOW – the D key. This slows the movement by about half. To return to the normal speed, press FAST – the F key.

To change the black square to a different character, press 9, then input the new character. So, to get a grey trail, press 9, shift graphic shift H, then press NEWLINE twice. To get zero,

press 9, 0 NEWLINE. To get a space for moving without a trail and rubbing out lines, just press 9, NEWLINE.

One more command is TEXT. When you wish to write something on screen, it is awkward to keep on pressing 9, the new letter, move along, and so on. By pressing T, you can input a group of letters/graphics, so to write 'HELLO' press T, HELLO then NEWLINE. If your string is too long, and won't fit on screen, you will lose the excess letters.

Notes on listing

10 goto variables.

100 Print the character in use, B$, making it flicker, so as to be seen more easily.

110 Wait for key to be pressed.

200–240 Movement of character. Note that the keys 1–4 appear in both X+Y co-ord change lines. 230–240 stop the character going offscreen, so that diagonal movement keys will produce vertical/horizontal movement when reaching the edge of the screen.

250–280 If in normal speed, repeat loop. If in slow speed stop for a moment. Increase F as desired, for longer pauses.

300–310 If the key showing SLOW or FAST is pressed, change to appropriate speed.

320 If key 9 is pressed, goto change graphic routine.

330 If key T is pressed, goto text routine.

400–430 Change graphic routine. Make sure new graphic is one character long.

500–540 Text routine. Input text, shorten it if too long for screen and put back cursor directly after the text.

8000–8030 Variables X+Y are co-ords of cursor, B$ is the cursor character, initially an inverse space, and SP is the speed which is initially fast.

Listing

```
   1 REM SCREEN ART 1
  10 GOSUB 8000
 100 PRINT AT Y,X;" ";AT Y,X;"█"
;AT Y,X;B$
 110 IF INKEY$="" THEN GOTO 100
```

```
 120 LET A$=INKEY$

 200 IF A$<"1" OR A$>"6" THEN GO
TO 300
 210 LET X=X+(A$="6")+(A$="2")+(
A$="3")-(A$="5")-(A$="1")-(A$="4
")
 220 LET Y=Y+(A$="6")+(A$="3")+(
A$="4")-(A$="7")-(A$="1")-(A$="2
")
 230 LET X=X+(X<0)-(X>31)
 240 LET Y=Y+(Y<0)-(Y>21)
 250 IF SP=1 THEN GOTO 100
 260 FOR F=1 TO 10
 270 NEXT F
 280 GOTO 100

 300 IF A$="D" THEN LET SP=0
 310 IF A$="F" THEN LET SP=1
 320 IF A$="9" THEN GOTO 400
 330 IF A$="T" THEN GOTO 500
 390 GOTO 250

 400 INPUT A$
 410 IF A$="" THEN LET A$=" "
 420 LET B$=A$(1)
 430 GOTO 100

 500 INPUT A$
 510 IF LEN A$+X>31 THEN LET A$=
A$( TO 31-X)
 520 PRINT AT Y,X;A$
 530 LET X=X+LEN A$
 540 GOTO 100

 8000 LET X=0
 8010 LET Y=21
 8020 LET B$="■"
 8030 LET SP=1
```

Screen Art 2

Listing occupies 1.8K
Program runs in 2.9K

To use

A number of new commands have been included. The most
important of these are PLOT and UNPLOT. This means that
you can draw things half-size (i.e., twice the resolution) easily.
Press PLOT – the Q key – to get into PLOT mode, and the

computer will draw narrow black lines instead of the normal thick ones. This is very useful for drawing delicate outlines and details. You cannot use TEXT while in PLOT mode, but you can move in all eight directions normally. To wipe out or move around, you press UNPLOT – the W key – which is equivalent to the space in PRINT mode, only one-quarter the size. (Note: moving on to a PRINTed graphic with a PLOTted point will erase the whole PRINTed graphic, unless it is a graphic which could be formed out of PLOTted points. To return to PRINT mode, just press PRINT – the P key.)

Another useful set of commands is the RETURN group. These are usable in both PRINT and PLOT mode. The RETURN commands the return of the cursor to one of nine positions which are the four corners, the centres of the four sides and the centre of the screen. They are numbered:

$$1 \quad 2 \quad 3$$
$$4 \quad 5 \quad 6$$
$$7 \quad 8 \quad 9$$

The cursor starts at position 7. To return to the top right-hand corner, press RETURN – the Y key, then press 9. To return to the centre, press RETURN, 5, and so on.

If you are lucky enough to have a ZX Printer then add these lines.

```
360 IF A$="Z" THEN GOSUB 550
550 INPUT Z$
560 IF Z$<>"Z" THEN RETURN
570 COPY
580 RETURN
```

These allow you to COPY your designs. Note that you have to press Z and Z NEWLINE to COPY, so that you cannot COPY by accident. Once the picture is COPYed, you can carry on with the same drawing. You can only COPY while in PRINT mode. This opens up many more possibilities, such as making personalized Christmas cards, ex-libris labels, and you can make a collection of your drawings.

Notes on listing

340 If RETURN is pressed goto RETURN sub-routine.

350 If PLOT or UNPLOT key is pressed, goto PLOT routine.

600–650 RETURN sub-routine. Co-ordinates of nine places to which you can RETURN are stored in C$.

700–850 Main PLOTting/UNPLOTting routine.

700–710 converts co-ordinates of PRINTed point to that of PLOTted point in same position.

720–730 Set PL according to whether PLOTting or UNPLOTting.

740–770 Make point flicker so it can be seen and PLOT or UNPLOT according to PL.

780–800 Check for key pressed, and divert to line 900 unless movement. This saves time.

810–850 Move point within screen boundaries and repeat loop.

900–990 If PLOT or UNPLOT changed, set PL accordingly. If RETURN or PRINT pressed, goto relevant sub-routines.

1000 PRINT routine. Change PLOTted co-ords back to PRINTed co-ords and go to main loop.

1100–1150 RETURN sub-routine. Make sure that key pressed is in correct range.

The co-ords of points to RETURN to are stored in C$.

Note that since these are PRINT co-ords, they have to be converted to PLOT words.

8040 C$ contains the co-ords of the nine RETURN points. Each co-ord is given two numbers and the Y co-ord is given first. The co-ordinate pairs are separated by full stops for ease of entering and checking.

Explanation of copy routine

This is straightforward, but note that your have to press Z then Z NEWLINE, so that you cannot accidentally COPY. Note also that you can continue the drawing after COPYing, which you cannot do by pressing BREAK COPY.

Listing

```
    1 REM SCREEN ART 2

  340 IF A$="Y" THEN GOTO 800
  350 IF A$="Q" OR A$="W" THEN GO
TO 700

  600 LET A$=INKEY$
  610 IF A$<"1" OR A$>"9" THEN GO
TO 600
  620 LET RE=VAL A$*5-4
  630 LET Y=VAL C$(RE TO RE+1)
  640 LET X=VAL C$(RE+2 TO RE+3)
  650 GOTO 100

  700 LET Y=43-2*Y
  710 LET X=2*X
  720 IF A$="Q" THEN LET PL=1
  730 IF A$="W" THEN LET PL=0
  740 PLOT X,Y
  750 UNPLOT X,Y
  760 IF PL=1 THEN PLOT X,Y
  770 IF PL=0 THEN UNPLOT X,Y
  780 IF INKEY$="" THEN GOTO 740
  790 LET A$=INKEY$
  800 IF A$<"1" OR A$>"8" THEN GO
TO 900
  810 LET X=X+(A$="8")+(A$="2")+(
A$="3")-(A$="5")-(A$="1")-(A$="4
")
```
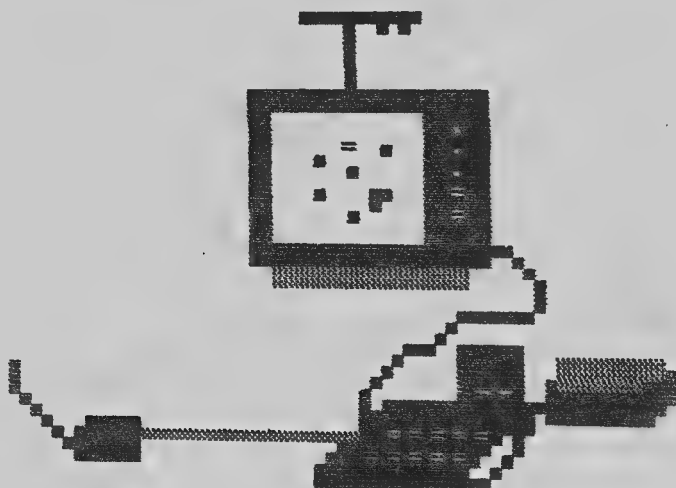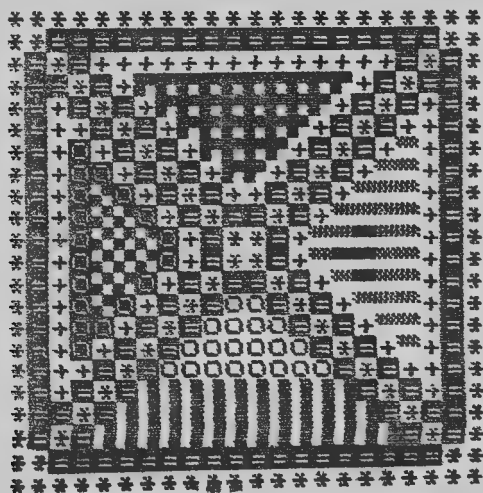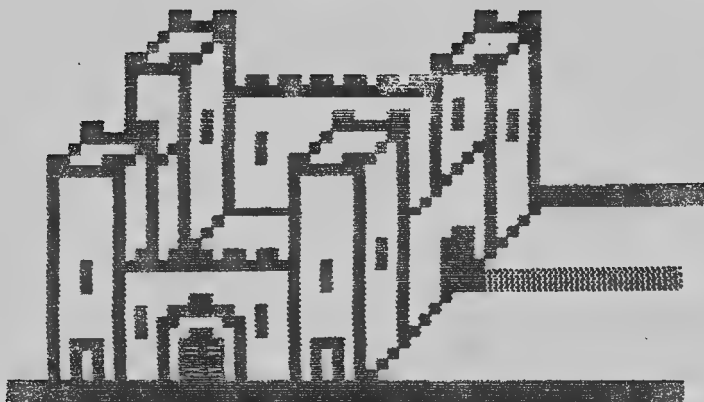
```
 820 LET Y=Y+(A$="7")+(A$="1")+(
A$="2")-(A$="6")-(A$="3")-(A$="4
")
 830 LET X=X+(X<0)-(X>63)
 840 LET Y=Y+(Y<0)-(Y>43)
 850 GOTO 730

 900 IF A$="W" THEN LET PL=0
```

```
 910 IF A$="O" THEN LET PL=1
 920 IF A$="P" THEN GOTO 1000
 930 IF A$="Y" THEN GOTO 1100
 990 GOTO 730

1000 LET Y=INT ((43-Y)/2)
1010 LET X=INT (X/2)
1020 GOTO 230

1100 LET A$=INKEY$
1110 IF A$<"1" OR A$>"9" THEN GO
TO 1100
1120 LET RE=VAL A$*5-4
1130 LET X=VAL C$(RE+2 TO RE+3)*
2
1140 LET Y=43-VAL C$(RE TO RE+1)
*2
1150 GOTO 730

8040 LET C$="0000.0015.0031.1000
.1015.1031.2100.2115.2131."
8050 RETURN.
```

Text Display

Program runs in 1K

To use

This program scrolls text continuously across the screen from right to left, with a double border scrolling the other way. It is suitable for short messages, notices and advertisements. Using 1K, the message can be up to about 80 characters long, with a simple border. With extra memory, you can have much longer messages, e.g., complete specifications of the computer.

When you run, you are first asked to input the text, then the border. The border can be made of any characters, and graphics look very effective.

Notes on listing

10–40 Input text. If length of text smaller than 31 characters, double it until it is 31 or more characters long. Remember to put a space after the text.

50–80 Repeat for border. Note for both text and border you must enter some character, even if it is only a space. Just pressing NEWLINE will result in an endless loop.

100 PRINT first 31 characters of text.

110 Put the first character of text at end, i.e., scroll right to left.

120–140 PRINT two borders.

150 Scroll border left to right.
160 Repeat until BREAK pressed.

Listing

```
   1 REM TEXT DISPLAY
   2 REM  BY S.R.SPEEL

  10 PRINT "TEXT?"
  20 INPUT A$
  30 IF LEN A$<31 THEN LET A$=A$
+A$
  40 IF LEN A$<31 THEN GOTO 30

  50 PRINT "BORDER?"
  60 INPUT B$
  70 IF LEN B$<31 THEN LET B$=B$
+B$
  80 IF LEN B$<31 THEN GOTO 70
  90 CLS

 100 PRINT AT 6,0;A$( TO 31)
 110 LET A$=A$(2 TO )+A$(1)
 120 FOR G=0 TO 1
 130 PRINT AT 3+G*6,0;B$( TO 31)
 140 NEXT G

 150 LET B$=B$(LEN B$)+B$( TO LE
N B$-1)
 160 GOTO 100
```
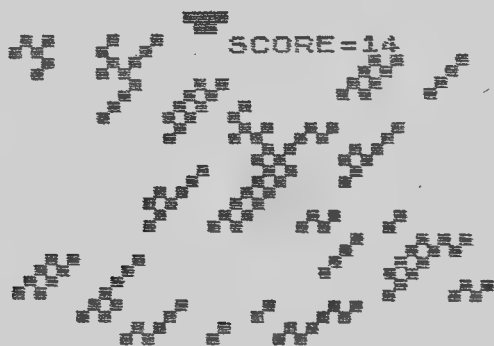
Avoid

Program runs in 1K

To use

You are in control of a small vehicle at the top of the screen and can move it left and right with keys 5 and 8. Your object is to find a way through the maze of blocks which rise constantly towards you. The game ends when you crash into a block from above and your score is then given. You can crash sideways into blocks without harm.

Notes on listing

10 X is the x co-ord of the object.

20 S is the score.

30 PRINT vehicle.

40–80 PEEK at two squares directly underneath vehicle. If either contains a graphic shift T (code 6) then goto end of program.

90 Move vehicle left or right according to key pressed. Make sure it does not go too far left or right.
100 PRINT blocks near bottom of screen. To make game easier, reduce number of block graphics from 6 to 4.
110 SCROLL blocks towards your vehicle.
120 Increase score.
130 Repeat until you crash.
140 You crash – score given.

Listing

```
   1 REM AVOID
   2 REM   BY S.R.SPEEL

  10 LET X=10
  20 LET S=0

  30 PRINT AT 0,X;"  � ";AT 1,X+
1;
  40 FOR F=1 TO 2
  50 LET A=PEEK (PEEK 16396+256*
PEEK 16399)
  60 PRINT " ".
  70 IF A=6 THEN GOTO 140
  80 NEXT F

  90 LET X=X+(INKEY$="8")-(INKEY
$="5")-(X>20)+(X<1)
 100 PRINT AT 15,RND*20;"▮▮ ▮";
AT 15,RND*20;"▮ ▮▮"
 110 SCROLL
 120 LET S=S+1
 130 GOTO 30

 140 PRINT "SCORE=";S
```

Adventure – Hero Maker

The idea behind Adventure games is that you represent some sort of warrior who goes in search of treasure, usually in a dungeon, castle or other enclosed area. You search through the rooms or caverns looking for gold, weapons and other useful things. Each treasure is generally guarded by some kind of monster which you have to kill. If you survive and get out, your success is measured in how much treasure you escaped with and how many monsters you have killed.

At each stage of the game you are told where you are, what is around you and, if there is a monster there, what type it is. You have a set of commands to move you through the cave system and to fight monsters.

One way of programming such a game is to have a pre-planned set-up. The program contains monsters, treasures and traps thought up by the programmer. These always seem exciting for the first couple of games – for instance you might be able to get past the pit in Room A using a plank from Room B and attack the 'ghosts' in Room D using the sword in Room C. However, once you have played the game a few times, you know all the traps, where the gold is hidden and how to maximize your chances of winning. In effect the game is 'used up'.

Another way of making an adventure is to use a random set-up. A system of caves or rooms is randomly produced and pre-defined monsters and treasure are placed at random in each cave. Although you will gradually learn the best tactics against a particular type of monster, you will not get to the stage where you can win every time, or know just what to expect. This is the system which I use for Hero Maker.

YOU ENTER THE CAVE SYSTEM BY
SLIDING DOWN A LONG CHUTE. THIS
OPENS INTO CAVE 1/3.

THIS CAVE CONTAINS A SMALL POOL
OF FRESH WATER.
THERE IS A WARTHOG IN THE
CAVE.
WHAT DO YOU DO?

YOU TRY TO STAB IT WITH YOUR
SWORD.
YOU HURT IT BADLY.
 THE MONSTER TRIES TO KICK YOU
BUT FAILS.
WHAT DO YOU DO?

YOU TRY TO SLICE IT WITH YOUR
AXE.
YOU HURT IT BADLY.
THE WARTHOG IS DEAD.
YOU FIND 3 GOLD COINS.
 AND A AXE
YOU DRINK FROM THE POOL.
3
 PRESS   R) TO REST
             T) TO SEE STATUS
             N) TO GO NORTH
             E) TO GO EAST
             W) TO GO WEST

Hero Maker Part 1

Listing occupies 9.0K
Program runs in 11.0K

Typing in the program
Adventure games are complex and take up a lot of RAM. As
you won't always have the time to type in many pages of listing
in one go, I have split up the program into stages. Put in the
main 'program core' and add the other stages later. At any
point, the game is playable, although the first couple of stages
are a little inflexible. When you feel ready to progress to some-
thing more complicated, add the next stage. You will find in
the end you have a very worthwhile game which will keep its
playing value.

The background. You are a young warrior who, although of
noble birth and great strength, are too poor to take your rightful
place among the country's heroes. To become an acknowledged
hero you must enter the Caverns of Power. This is a large cave
system, originally used by men, but long since overrun by cruel
and ferocious monsters, guarding their treasure to the death.
Once in, there is no turning back, you must fight your way to
the only exit . . . .

The Core Program
Listing occupies 6.1K
Program runs in 7.3K

To use
In the core program, the main cave system is set up and
monsters placed. This means that when you RUN the program
the screen will be blank for a few seconds. You are then
launched straight into your first fight. You have two weapons:

a sword and an axe. When, during a fight, you are asked 'what do you do?', press key 1 to use your sword and key 3 to use your axe. Later you will find other weapons. A club is used with key 2, a heatwand with key 4 and an icestaff with key 5.

Only experience will teach you how best to kill monsters, but some methods are obvious. For example, a furred lion will be cold-resistant (against icestaffs) but its fur will easily be burnt by a heatwand.

When you fight a monster, you are told if you have hit or wounded it badly. You are also told your resistance. This number tells you how many more hits you can take. You start with resistance 10, and if it goes to 0, you 'die'. When monsters bite or otherwise damage you, your resistance goes down.

When you kill a monster, you find treasure in the form of gold pieces, and one new weapon. You can then rest for as long as you wish. Each time you rest, your resistance increases until up to its maximum. Your 'maximum possible resistance' also increases for each monster killed, to an absolute maximum of around 20.

The cave system is a grid of 10 by 5 cave locations. Your cave position is given as 2 numbers separated by a line. The first is the cave's north co-ord, the second its east co-ord. So cave 1/3 – where you enter – is in the centre of the southernmost row. You can escape by going north out of one of the caves in row 10, but you have to discover which one. In each row, one cave location is actually solid rock, so you will not usually be able to go in a straight line to the exit.

Your aim is to get the highest 'hero rating', which is made up from your 'heroic bonus' and how much money you have accumulated.

Notes on listing
The basic core program includes the complete fighting system, which is very complex. Each type of monster has its own individual resistance to each type of weapon.

The monsters attack in different ways, and are better at some

forms of attack than others, e.g., a cave hyena is good at biting (bite value = 5) but cannot claw or crush at all (claw value and crush value = 0). Other factors, such as the fact that you are gradually becoming better at using a weapon, are also taken into account. However, the computer deals with all this, and you need not know how it works when playing the game.

10 GOSUB variables and set up cave system.

1000–1010 Pairs of lines like this, setting SN to some number and then GOSUBing 6500 or 6550 occur throughout the program. GOSUB 6550 means SCROLL SN times, GOSUB 6500 means PAUSE for a couple of seconds, then SCROLL SN times.

1020 Check for winning – you exit cave system.

1030–1100 You enter cave. If monster there, goto 2000.

1110 No monster in cave.

1200–1550 Possible choices of action given, and commands accepted. Checks for wrong commands and ignores them. Carries out correct commands.

2000–2020 Select random monster. M$ and P$ are parts of the monster array, and are more convenient to use.

2100–2150 Wait for your fighting commands. If no command, go straight on to monster's attack, as you are assumed to be standing around doing nothing.

2200–2280 Monster selects its attack, with a bias towards what it can do best.

4000–4360 You hit monster, damage calculated. Death of monster checked for. Lines 4230–4260 see how many times you have used a weapon, and whether you gain experience in using it.

4500–4590 Monster dies. If treasure, goto 7000. Increase number of monsters killed, maximum resistance (if appropriate) and heroic bonus.

4900–4910 You press a wrong key. Note that you lose your chance to strike that go, due to indecisiveness.

5000–5230 Monster strikes you. Your resistance is decreased if it hits you successfully, and your demise checked for.

5500–5590 You win. Your final hero rating is the sum of your heroic bonus (which depends on how tough your fights with the monsters were) and your gold pieces.

6500–6530 PAUSE routine.

6550–6580 SCROLL routine.

7000–7070 You find treasure, gold pieces and one weapon.

8000–9990 Set up variables and caves. This is done in FAST mode to save time.

8010–8060 Your possible weapons. Each weapon has an action, e.g., STAB, which is 6 letters minus VAL X$(F,7), and a name, e.g., SWORD, which is 8 letters long minus VAL X$(F,16). The last figure is the maximum damage possible with that weapon.

8200–8240 Directions. The figures are the co-ords to move your position north and east (– 1 for south and west).

8300–8350 Monsters. The figures are as follows (taking the unicorn as an example): M$(1,1 TO 12) = name, M$(1,13) = number of characters to remove from end of name, M$(1,14 TO 15) = resistance of monster, M$(1,16 TO 20) = defence values against sword, club, axe, heatwand and ice-staff respectively. M$(1,21 TO 25) = attack values for bite, kick, impale, claw and crush respectively so that the unicorn is good at impaling (value 6) and biting (value 3) but cannot claw or crush (value 0). M$(1,26) is the extra cash bonus for killing that monster, 0 for the unicorn as it does not collect extra gold. M$(1,27) is the heroic bonus for killing that monster, the unicorn's value of 5 means that it is moderately difficult to kill.

9000–9260 Set up R$ which holds the cave system. Note that the cave system is actually a 7 by 12 block of caves, of which only the middle 5 by 10 are used. This is to stop the computer looking outside the array when considering exits from a cave.

9300–9370 Variables. RS = your initial resistance, CASH = gold pieces gained, MK = monsters killed, HQ = heroic bonus, W$ = your weapons, Y$ = your current max resistance and experience bonus, CEX = extra cash for killing monster, Y = how many times you have used each weapon successfully. CVN and CVE are your co-ords

in the cave system.

9800–9940 Short introduction, SLOW and start enter cave routine.

Listing for Core 1

```
   1 REM HERO MAKER
   2 REM    COPYRIGHT 1982
            S.ROBERT SPEEL
   3 REM    CORE PROGRAM

  10 GOSUB 8000

1000 LET SN=18
1010 GOSUB 6550
1020 IF CVN=12 THEN GOTO 5500
1030 PRINT AT 0,0;"YOU ENTER CAV
E ";CVN-1;"/";CVE-1;AT 1,0

1100 IF R$(CVN,CVE,1)="1" THEN G
OTO 2000
1110 PRINT " THE CAVE CONTAINS N
O MONSTERS."
1200 PRINT " PRESS";TAB 10;"R) T
O REST"
1210 FOR F=1 TO 4
1220 IF R$(CVN+VAL C$(F,6 TO 7),
CVE+VAL C$(F,8 TO 9),1)<"9" THEN
 PRINT TAB 10;C$(F,1);") TO GO "
;C$(F,1 TO 5)
1230 NEXT F

1250 FOR F=1 TO 400
1260 LET A$=INKEY$
1270 IF A$="R" THEN GOTO 1410
1280 IF A$="T" THEN GOTO 6000

1300 FOR G=1 TO 4
1310 IF R$(CVN+VAL C$(G,6 TO 7),
CVE+VAL C$(G,8 TO 9),1)<"9" AND
A$=C$(G,1) THEN GOTO 1370
1320 NEXT G
1330 NEXT F
1360 GOTO 1250

1370 LET CVN=CVN+VAL C$(G,6 TO 7
)
1380 LET CVE=CVE+VAL C$(G,8 TO 9
)
1390 PRINT " YOU DECIDE TO GO ";
C$(G,1 TO 5);"..."
```

```
1400 GOTO 1000
1410 IF RS<VAL Y$(1 TO 2) THEN L
ET RS=RS+INT (RND*5)+1
1420 IF RS>VAL Y$(1 TO 2) THEN L
ET RS=VAL Y$(1 TO 2)
1430 PRINT AT 16,1;"YOU HAVE A R
EST...","YOUR RESISTANCE IS NOW
";RS
1460 FOR F=1 TO 50
1470 NEXT F
1510 FOR F=16 TO 21
1520 PRINT AT F,0;"

"
1530 NEXT F
1540 PRINT AT 16,0;
1550 GOTO 1250

2000 LET N$=M$(INT (RND*5)+1)
2010 LET P$=N$( TO 12-VAL N$(13)
)
2020 PRINT "THERE IS A ";P$;" IN
 THE";TAB 0;" CAVE."

2100 PRINT "WHAT DO YOU DO?"
2110 LET Z=0
2120 FOR F=1 TO 200
2130 LET A$=INKEY$
2140 IF A$>"0" AND A$<"6" THEN L
ET Z=VAL A$
2150 IF Z THEN GOTO 4000
2160 NEXT F
2170 LET SN=10
2180 GOSUB 6550

2200 LET MY=INT (RND*(VAL N$(21)
+VAL N$(22)+VAL N$(23)+VAL N$(24
)+VAL N$(25)+1))
2210 LET MX=0
2250 FOR F=1 TO 5
2260 LET MX=MX+VAL N$(F+20)
2270 IF MY<=MX THEN GOTO 5000
2280 NEXT F

4000 LET SN=12
4010 GOSUB 6550
4020 IF U$(Z)="0" THEN GOTO 4900
4030 PRINT "YOU TRY TO ";X$(Z, T
O 6-VAL X$(Z,7));" IT WITH YOUR"
;TAB 0;X$(Z,8 TO 15-VAL X$(Z,16)
);"."
4100 LET HR=INT (RND*VAL X$(Z,17
))+VAL Y$(Z+2)+1
```

```
4110 IF HR>VAL N$(15+Z) THEN GOT
O 4200
4120 PRINT "YOU DO NOT HURT IT."
4130 GOTO 2200

4200 LET DAM=HR-VAL N$(15+Z)
4210 PRINT ("YOU WOUND IT." AND
DAM<3);("YOU HURT IT BADLY." AND
 DAM>2)
4230 LET Y(Z)=Y(Z)+1
4240 IF Y(Z)<10 THEN GOTO 4300
4250 LET Y(Z)=0
4260 IF Y$(Z+2)<"3" THEN LET Y$(
Z+2)=STR$ (VAL Y$(Z+2)+1)
4300 LET N$(14 TO 15)=STR$ (VAL
N$(14 TO 15)-DAM)
4310 IF VAL N$(14 TO 15)<1 THEN
GOTO 4500
4320 IF VAL N$(14 TO 15)<3 THEN
PRINT "THE ";P$;" IS VERY WEAK."
4360 GOTO 2200

4500 PRINT "THE ";P$;" IS DEAD."
4520 LET R$(CVN,CVE,1)="0"
4530 LET CEX=VAL N$(26)
4550 IF Y$(1 TO 2)<"20" THEN LET
 Y$(1 TO 2)=STR$ (VAL Y$(1 TO 2)
+INT (RND*2)+1)
4560 LET MK=MK+1
4570 IF R$(CVN,CVE,3)>"0" THEN G
OSUB 7000
4580 LET HQ=HQ+VAL N$(27)
4590 GOTO 1120

4900 PRINT " YOU TRY TO USE A WE
APON WHICH  YOU DO NOT HAVE."
4910 GOTO 2200

5000 LET G$=F$(F)
5010 PRINT " THE MONSTER TRIES T
O ";G$( TO 7-VAL G$(8));" YOU"
5100 LET MD=INT (RND*10)-10+VAL
N$(F+20)
5110 IF MD>=0 THEN GOTO 5200
5120 PRINT "BUT FAILS."
5140 GOTO 2100
5200 LET RS=RS-MD-1
5210 PRINT "YOU ARE HURT, YOUR R
ESISTANCE IS NOW ";RS;(". YOU DI
E." AND RS<1)
5220 IF RS<1 THEN STOP
5230 GOTO 5130
```

```
5500 PRINT "████████████████████
████████████"
5510 FOR F=1 TO 20
5520 PRINT "█";TAB 31;"█"
5530 NEXT F
5540 PRINT "████████████████████
████████████"
5550 PRINT AT 3,3;"YOU HAVE ESCA
PED ALIVE.";AT 5,4;"YOU HAVE ";C
ASH;" GOLD COINS,";AT 8,3;"AND H
AVE KILLED ";MK;" MONSTERS."
5560 PRINT AT 10,4;"YOUR HEROIC
BONUS IS ";HQ
5580 PRINT AT 12,4;"YOUR HERO RA
TING IS ";HQ+CASH
5590 STOP

6500 FOR F=1 TO 50
6530 NEXT F

6550 FOR F=1 TO SN
6560 SCROLL
6570 NEXT F
6580 PRINT AT 0,0;
6590 RETURN

7000 LET R$(CVN,CVE,3)="0"
7010 LET CD=INT (RND*15)+2+CEX
7030 LET CASH=CASH+CD
7040 LET WF=INT (RND*5)+1
7050 LET W$(WF)=STR$ (VAL W$(WF)
+1)
7060 PRINT "YOU FIND ";CD;" GOLD
 COINS";TAB 0;" AND A ";X$(WF,8
TO 15-VAL X$(WF,16))
7070 RETURN

8000 FAST
8010 DIM X$(5,17)
8020 LET X$(1)="STAB..2SWORD...3
6"
8030 LET X$(2)="BASH..2CLUB....4
7"
8040 LET X$(3)="SLICE.1AXE.....5
5"
8050 LET X$(4)="BURN..2HEATWAND0
8"
8060 LET X$(5)="FREEZE0ICESTAFF0
9"

8100 DIM F$(5,8)
8110 LET F$(1)="BITE...3"
8120 LET F$(2)="KICK...3"
```

```
8130 LET F$(3)="IMPALE.1"
8140 LET F$(4)="CLAW...3"
8150 LET F$(5)="CRUSH..2"

8200 DIM C$(4,9)
8210 LET C$(1)="NORTH0100"
8220 LET C$(2)="SOUTH-100"
8230 LET C$(3)="EAST 0001"
8240 LET C$(4)="WEST 00-1"

8300 DIM M$(5,27)
8310 LET M$(1)="UNICORN......5143
45533460005"
8320 LET M$(2)="WARTHOG.....5102
41352560003"
8330 LET M$(3)="FURRED LION.1091
24068005025"
8340 LET M$(4)="CAVE HYENA..2083
24555002013"
8350 LET M$(5)="GORILLA.....5143
52165100825"

9000 RAND
9010 DIM R$(12,7,3)
9020 FOR F=2 TO 11
9030 LET R$(F,1,1)="9"
9040 LET R$(F,7,1)="9"
9050 FOR G=2 TO 6
9060 LET R$(F,G,1)="1"
9080 LET R$(F,G,3)="1"
9090 NEXT G
9100 LET R$(F,INT (RND*5)+2,1)="
9"
9110 NEXT F
9200 FOR F=1 TO 7
9210 LET R$(1,F,1)="9"
9220 LET R$(12,F,1)="9"
9230 NEXT F
9240 LET RE=INT (RND*3)+3
9250 LET R$(12,RE,1)="1"
9260 LET R$(11,RE,1)="1"

9300 LET RS=10
9310 LET CASH=0
9320 LET MK=0
9330 LET HQ=0
9340 LET W$="10100"
9350 LET Y$="1010100"
9360 LET CEX=0
9370 DIM Y(5)
9700 LET CVN=2
9710 LET CVE=4
```

```
9800 PRINT " YOU ENTER THE CAVE
SYSTEM BY    SLIDING DOWN A DEEP
CHASM. THIS OPENS INTO CAVE 1/3.
"
9980 SLOW
9990 GOTO 1040
```

Hero Maker Part 2
Listing occupies 7.7K
Program runs in 9.0K

To use
When you are ready to expand Hero Maker, add this
expansion. Don't worry if a few of the new lines expunge old
ones – this is necessary to update the listing to cope with the
new parts.

This will expand the fighting possibilities, so for the first time
the cave will have a short description and some effect on
combat. Caves can now have pools of water, or be slippery-
floored, or be filled with steam.

You can also look up your status. This gives up-to-date
details on your resistance, weapons and gold pieces. It also
gives your heroic bonus, the measure of your success in killing
monsters.

```
▰▰▰▰ STATUS ▰▰▰▰

YOUR RESISTANCE IS 19
YOU HAVE 38 GOLD PIECES
   YOUR WEAPONS ARE:-
            1 SWORD
            1 CLUB
            2 HEATWANDS
            1 ICESTAFF
HEROIC BONUS = 22
 (PRESS NEWLINE TO CONTINUE)
```

You can now break your weapons, so you will be glad for a
few spares. You can also be interrupted by monsters which just
wander in while you are resting. These monsters have no
treasure of course.

Due to the increased difficulty of killing monsters, your

initial resistance has been increased to 20, and the maximum
to 30.

Notes on listing

1040–1050 SLP is the slip value and VIS the visibility value for
a cave.

1090 GOSUB cave's description.

1200–1280 A status command has been added.

1340–1500 These lines make it possible for a monster to just
wander into the cave where you are, while you are resting or
thinking.

2400–2430 Room with pool. At this stage, this only means you
might slip and fall while fighting.

2500–2520 Slimy room – easy to fall over here.

2600–2620 Steam room – your view of the monster may be
obscured here.

3000–3080 Enter wandering monster.

4050–4060 Check for falling over and steam. Note that you can
only slip when using a heavy weapon such as a sword, axe
or, worst of all, a club. Light weapons – heatwand and ice-
staff – will not make you slip.

4150–4160 You miss due to steam obscuring your sight.

4330–4350 Your weapon breaks.

4600–4660 You slip and fall over. Note you can actually
damage yourself like this.

6000–6170 Status. Resistance, gold pieces, weapons and
heroic bonus given.

9070 Each cave is given a value to determine if it is a steam
room, pool room, etc.

9300–9350 New resistance = 20.

Listing

```
   3 REM CORE + PART 2

1040 LET SLP=0
1050 LET VIS=0
1090 GOSUB 2300+100*VAL R$(CUN,C
UE,2)
```

```
1200 PRINT " PRESS";TAB 10;"R) T
O REST";TAB 10;"T) TO SEE STATUS
"
1280 IF A$="T" THEN GOTO 6000

1340 IF RND<.1 THEN GOTO 3000
1500 IF RND<.1 THEN GOTO 3000

2400 PRINT "THIS CAVE CONTAINS A
 SMALL POOL OF FRESH WATER."
2410 LET SLP=.2
2420 LET WA=1
2430 RETURN

2500 PRINT " THE GROUND HERE IS
COVERED IN  SLIMY ALGAE- WATCH Y
OUR BALANCE."
2510 LET SLP=1
2520 RETURN

2600 PRINT " THIS CAVE IS HOT AN
D FULL OF   STEAM."
2610 LET VIS=1
2620 RETURN

3000 LET N$=M$(INT (RND*5)+1)
3010 LET P$=N$( TO 12-VAL N$(13)
)
3020 PRINT " SUDDENLY, A ";P$;"
COMES";TAB 2;"IN."
3050 LET SN=20
3060 GOSUB 6500
3080 GOTO 2100

4050 IF (Z=1 AND RND<.1*SLP) OR
(Z=2 AND RND<.2*SLP) OR (Z=3 AND
 RND<.15*SLP) THEN GOTO 4500
4060 IF RND<VIS*.1 THEN GOTO 415
0

4150 PRINT "DUE TO THE THICK STE
AM, YOUR AIM IS SPOILT, AND YOU
MISS."
4160 GOTO 2200

4330 IF RND<.92 THEN GOTO 2200
4340 PRINT " UNLUCKILY, YOUR ";X
$(Z,8 TO 15-VAL X$(Z,16));" BREA
KS."
4350 LET W$(Z)=STR$ (VAL W$(Z)-1
)
```

```
4550 IF Y$(1 TO 2)<"30" THEN LET
 Y$(1 TO 2)=STR$ (VAL Y$(1 TO 2)
+INT (RND*2)+1)

4600 PRINT " YOU SLIP AND FALL O
VER, MISSING TOTALLY."
4610 IF RND<.0 THEN GOTO 2200
4620 LET RS=RS-INT (RND*3)-1
4630 PRINT "YOU ARE BRUISED, AND
 YOUR";TAB 0;"RESISTANCE IS NOW
";RS
4650 IF RS<1 THEN GOTO 5300
4660 GOTO 2200

6000 LET SN=15
6010 GOSUB 6550
6020 PRINT "***** STATUS *****";
TAB 1;TAB 0;"YOUR RESISTANCE IS
";RS;TAB 0;"YOU HAVE ";CASH;" GO
LD PIECES";TAB 2;"YOUR WEAPONS A
RE:-"
6030 FOR F=1 TO 5
6040 IF VAL W$(F)>0 THEN PRINT T
AB 7;W$(F);" ";X$(F,8 TO 15-VAL
X$(F,16));("S" AND VAL W$(F)>1)
6050 NEXT F

6100 PRINT "HEROIC BONUS = ";HO
6130 PRINT "(PRESS NEWLINE TO CO
NTINUE)"
6140 IF INKEY$="" THEN GOTO 6140
6150 LET SN=16
6160 GOSUB 6550
6170 GOTO 1200

9070 LET R$(F,G,2)=STR$ INT (RND
*4)

9300 LET RS=20
9350 LET Y$="2010100"

9380 LET TM=0
9390 LET TT=0
```

Hero Maker Part 3

Listing occupies 9.0K
Program runs in 11.0K

To use
Now that you are expert at Hero Maker, you will survive longer
in the caves. Therefore, we must consider eating and drinking.
While the typical adventurer carries food, water is more bulky
and needed more often. You start off well watered, but
gradually you will become more and more thirsty until you die
of dehydration after eighteen hours. As each rest takes one
hour, and a fight or moving from one cave to another takes ten
minutes, you would not last long were it not for the caves with
pools. Up until now, these have not been very important, but
now you can drink from the pools and last another eighteen
hours. Just rest in a pool cave and you automatically drink. So,
as long as you go to a pool cave every few hours, you will be
OK.

   There is a new type of cave – the stalactite cave. Here stalac-
tites hang from the ceiling and fall off and may hit you. You
will have found that monsters do not slip or get bewildered by
steam, as they are used to their caves. However, no monster can
get used to falling stalactites, and they are just as likely to fall
on the monsters as on you.

   There are five new monsters. The death turtle is very well
armoured and the earth viper is rather vicious as, although it
has a low resistance, just a few bites from it can kill you.

Notes on listing
1060 DA is the chance of a stalactite falling on you. WA is set
         to 1 if there is drinking water (a pool) in the cave. TM is the
         time since you last drank (in 1/6 hours).
1440–1490 While you rest, a stalactite may fall on you (if there
         are any). You rest for one hour and if there is a pool of water,
         you drink.
1700–1860 Stalactite routine. The stalactite can hit you, the
         monster, or nothing at all. DM is the damage done by a

falling stalactite if it hits the monster. A falling stalactite reduces resistance by between 1 and 5.

2000 There are now ten monsters to choose from.

2050–2060 Make monsters weak for the first couple of combats, and stronger than usual for later combats (after thirty-six hours).

2300 Stalactite cave description.

3000–3050 Some adjustments to the wandering monster routine. Some SCROLLing is needed to prevent program crashing due to 'OUT OF SCREEN' problems.

5570–5580 Modification to 'you escape' routine.

6110–6120 Status now includes details of how long you have been in the caves, and when you had your last drink.

6540–6660 During routine, check if you die of thirst.

8300–8400 Increasing the size of M\$ and adding just five extra program lines gives five new monsters. This is the advantage of using string arrays.

Listing

```
   3 REM CORE + PARTS 2+3

1060 LET DA=0
1070 LET WA=0
1080 LET TM=TM+1

1350 LET TM=TM+1
1400 GOTO 1000
1440 IF RND<.1*DA THEN GOSUB 171
0
1450 IF TM>72 THEN PRINT "YOU AR
E VERY THIRSTY."
1480 LET TM=TM+6
1490 IF WA>0 THEN GOSUB 1600

1600 PRINT "YOU DRINK FROM THE P
OOL"
1610 LET TT=TT+TM/6
1620 LET TM=0
1630 RETURN

1700 IF RND<1/3 THEN GOTO 1800
1710 PRINT "A STALECTITE IS DISL
ODGED AND   CRASHES DOWN, ";
1720 IF RND<.5 THEN GOTO 1750
```

```
1730 PRINT "DOING NO HARM."
1740 RETURN
1750 PRINT "HITTING YOU."
1760 LET RS=RS-INT (RND*5)-1
1770 PRINT "YOUR RESISTANCE IS N
OW ";RS;(" YOU DIE INSTANTLY." A
ND RS<1)
1780 IF RS<1 THEN STOP
1790 RETURN

1800 PRINT " A STALECTITE FALLS
ON TOP OF   THE ";P$;
1810 LET DM=INT (RND*5)+1
1820 IF DM<3 THEN PRINT " CUTTIN
G IT."
1830 IF DM>2 THEN PRINT " AND IM
PALES IT."
1840 LET N$(14 TO 15)=STR$ (VAL
N$(14 TO 15)-DM)
1850 IF VAL N$(14 TO 15)<1 THEN
GOTO 4500
1860 RETURN

2000 LET N$=M$(INT (RND*10)+1)
2050 IF TT+TM/6<2 THEN LET N$(16
 TO 20)="11111"
2060 IF TT+TM/6>2 THEN LET N$(16
 TO 20)=STR$ (VAL N$(15 TO 19)+1
1111)
2070 IF TM>72 THEN LET RS=RS-2

2300 PRINT " THERE ARE STALECTIT
ES LOOSELY  ATTACHED TO THE ROOF
 OF THIS    CAVE."
2310 LET DA=2
2320 RETURN

3000 LET N$=M$(INT (RND*10)+1)
3020 SCROLL
3030 SCROLL
3040 PRINT AT 20,0;" SUDDENLY, A
";P$;" COMES";TAB 2;"IN."
3050 LET SN=22

4510 LET TM=TM+1
4540 IF WA=1 THEN GOSUB 1600

5130 IF RND<.2*DA THEN GOSUB 170
0

5570 PRINT AT 11,2;"YOU TOOK ";I
NT (TT+TM/6+.5);" HOURS TO ESCAP
E."
```

```
6580 PRINT AT 14,4;"YOUR HERO RA
TING IS ";HO+CASH

6110 PRINT " YOU HAVE BEEN IN TH
E CAVES FOR ";INT (TT+TM/6+.5);"
 HOURS,"
6120 PRINT "AND YOU LAST DRANK "
;INT (TM/6+.5);" HOURS AGO."

6540 IF TM>72 THEN PRINT "YOU HA
VE NOT HAD WATER FOR A      LONG T
IME."
6590 IF TM>108 THEN GOTO 9650
6600 RETURN

6650 PRINT "YOU DIE FROM LACK OF
 WATER."
6660 STOP

8300 DIM M$(10,27)
8360 LET M$(6)="CAVE PYTHON.1122
33615000846"
8370 LET M$(7)="CAVERN CROC.1076
43126002013"
8380 LET M$(8)="DEATH-TURTLE0187
53222001038"
8390 LET M$(9)="EARTH-VIPER.1043
13439000035"
8400 LET M$(10)="CAVE BEAR...308
573394004555"

9980 SLOW
```

Hero Maker   Conversion to Spectrum

Hero Maker is easy to convert for use on the Spectrum as there
is no graphic display. As it is a very long program you may have
difficulty in fitting the complete version on to a 16K Spectrum,
as 6.7K is used for the screen display. To condense it, use of
multi-statement lines is a good idea.

The SCROLL routine will have to be changed, using POKE
23692,255.

You can have a screen which does not clear, but continuously

scrolls to show the next bit of information. This will mean changing some of the PRINT AT statements.

I do not think that there is anything to be gained by changing the string arrays to DATA statements, as this would mean considerable reworking of the program.

One other change involves pauses. To avoid screen flicker, dummy FOR-NEXT loops have been used, and the number of loops needs to be increased on the Spectrum as it works faster. Alternatively, you could use PAUSE as the Spectrum does not flicker.

Fairway

Program occupies 2.7K
Program runs in 3.6K

To play
The computer asks whether you wish to start, or see a condensed version of the rules.

When you start the game, a display of part of the course appears, with your ball in the bottom left, a square hole and patches of 'grass' consisting of grey patches and groups of asterisks (*). You 'hit' the ball by typing in the initial of the direction you wish to go: N, S, W, E, NE, SE, NW, SW, followed by the distance you wish the ball to go.

Thus, to go north 20m, you type N20; to go south-east 45m, you type SE45.

Note that there are no spaces in the INPUT.

The field measures about 30m by 20m. However, there is a variable element in the strength of hitting – so it is not that easy!

The grey grass is very thick and the ball loses momentum when it goes into it and doesn't go so far.

The asterisks represent springy grass and your ball may bounce away in any direction, including straight back from this grass.

The ball also has to *end* in the hole; it will go straight over if you hit too hard.

When you get the ball into the hole, your number of shots is given. There are six holes in the course and the lower the final score is the better.

Notes on listing

10 GOSUB stating variables and rules.

80 GOSUB new hole, field, etc.

90–100 PRINT ball and hole.

110–120 Reset move and input it.

130–180 Set X1 and Y1 (increase in X+Y each go) according to inputted value.

190–200 Find how far ball meant to move. Make sure not more than 60.

210 Reset V, pointer to next character in A$.

220–430 Main loop.

   230 Print space where ball was.

   240–250 If 'grass' should be there, print it if 'thick', or goto sub-routine for 'springy' grass.

   260–270 If ball at edge of field, make it bounce back.

   280–290 Find new values of X+Y, making sure they are within limits of field.

   300–310 PEEK into ball's new position to see if grass there.

   320 If 'tough' grass, slow ball down.

   330 Print ball in new position.

   340 Do again until shot finished.

   350 Add 1 to 'shots' counter.

   400 If ball ended up in hole, goto finished-hole routine.

   410–430 Reset X1+Y1 and do another shot.

440–490 Message on finishing 1 hole.

   Wait for short time (increase if desired).

500–510 Increment hole counter and find total score so far.

520 If all holes completed, goto end.

530 New hole and goto new hole sub-routine, etc.

550–590 End of game, total score given, new game offered.

600–630 If ball in 'springy' grass then make ball go in random direction and put back grass when ball leaves.

5000–5090 Print eight sections of 'tough' grass and eight sections 'springy' grass at random positions on screen.

5100–5120 Print hole at random position but not too near ball's starting position. Note hole cannot be covered by grass.

5130–5180 Set variables. H = current hole, SC = number of shots for that hole, X,Y are co-ords of ball, X1+Y1 are the

N/S and W/E velocities respectively.

8000 Start of game.

8000–8030 Initial variables. H = number of holes so far, SCO = total score, V = pointer to inputted shot, L= what is underneath ball – initially a space.

8200–8250 Check if rules desired. Note that no flickers occur, as would occur if PAUSE were used.

8300–8440 Rules. Modify, increase or omit as desired.

For a full 18-hole course, change line 520 to 520 IF H > 17 THEN GOTO 550 and alter line 8130.
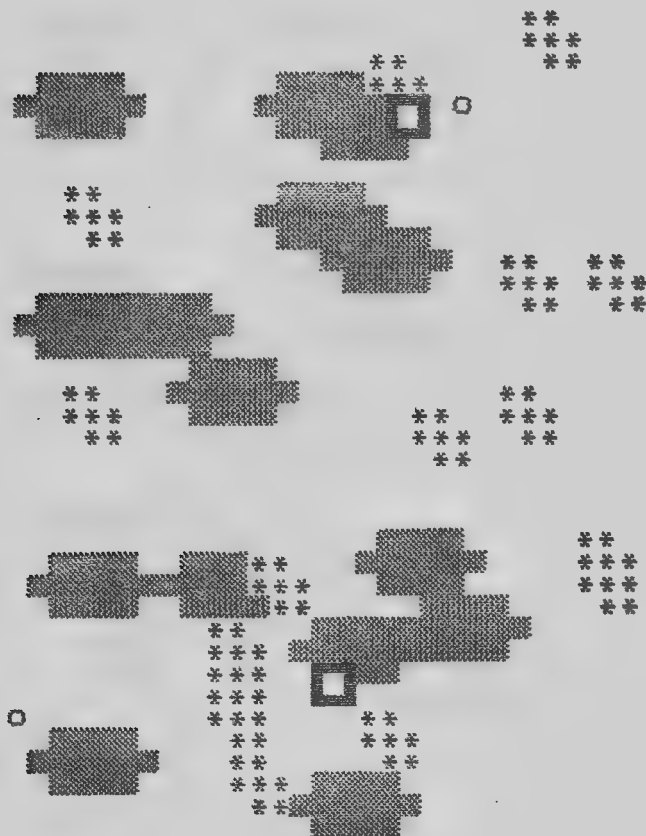
Listing

```
   1 REM          FAIRWAY
   2 REM      COPYRIGHT 1982
              S.ROBERT SPEEL
  10 GOSUB 8000
  20 CLS
  80 GOSUB 5000
  90 PRINT AT Y,X;"O";
 100 PRINT AT N,M;"██";AT N+1,M;
"██"
 110 DIM A$(4)
 120 INPUT A$
 130 LET Y1=(A$(1)="S")-(A$(1)="
N")
 150 LET V=V+(Y1<>0)
 160 LET X1=(A$(V)="E")-(A$(V)="
W")
 180 LET V=V+(X1<>0)
 190 LET D=VAL A$(V TO )+INT (RN
D*3)-1
 200 IF D>60 THEN LET D=60
 210 LET V=1

 220 FOR F=1 TO D+(D=0)
 230 PRINT AT Y,X;" "
 240 IF L=136 OR L=8 THEN PRINT
AT Y,X;"█"
 250 IF L=23 THEN GOSUB 600
 260 IF Y>20 OR Y<1 THEN LET Y1=
-Y1
 270 IF X<1 OR X>30 THEN LET X1=
-X1
 280 LET Y=Y+Y1+(Y=0)-(Y=21)
 290 LET X=X+X1+(X=0)-(X=32)

 300 PRINT AT Y,X;
 310 LET L=PEEK (PEEK 16398+256*
PEEK 16399)
 320 IF L=136 OR L=8 THEN LET F=
F+3
 330 PRINT AT Y,X;"O"
 340 NEXT F
 350 LET SC=SC+1
 400 IF (X=M OR X=M+1) AND (Y=N
OR Y=N+1) THEN GOTO 440
 410 LET X1=0
 420 LET Y1=0
 430 GOTO 100

 440 CLS
 450 IF SC<3 THEN PRINT "CONGRAT
ULATIONS";
 460 IF SC>6 THEN PRINT "UGH,";
```

```
 470 PRINT "  HOLE IN ";SC
 480 FOR F=1 TO 50
 490 NEXT F
 500 LET H=H+1
 510 LET SCO=SCO+SC
 520 IF H>5 THEN GOTO 550
 530 GOTO 20
 550 PRINT AT 10,1;"THATS IT.YOU
R TOTAL SCORE IS ";SCO
 560 PRINT AT 14,4;"ANOTHER GAME
? (Y/N)"
 570 IF INKEY$="Y" THEN RUN
 580 IF INKEY$="N" THEN STOP
 590 GOTO 570

 600 LET X1=INT (RND*3)-1
 610 LET Y1=INT (RND*3)-1
 620 PRINT AT Y,X;"*"
 630 RETURN
5000 FOR F=1 TO 6
5010 LET YC=RND*16+1
5020 LET XC=RND*20+2
5030 PRINT AT YC,XC;"▩▩▩▩";AT YC
+1,XC-1;"▩▩▩▩▩▩";AT YC+2,XC;"▩▩▩
▩"
5040 NEXT F
5050 FOR F=1 TO 6
5060 LET YC=RND*18
5070 LET XC=RND*26
5080 PRINT AT YC,XC;"**";AT YC+1
,XC;"***";AT YC+2,XC+1;"**"
5090 NEXT F
5100 LET M=INT (RND*20)+10
5110 LET N=INT (RND*15)
5120 PRINT AT N,M;"▩▩";AT N+1,M;
"▩▩"
5130 LET SC=0
5140 LET X=1
5150 LET Y=21
5160 LET X1=0
5170 LET Y1=0
5180 RETURN

8000 LET H=0
8010 LET SCO=0
8020 LET V=1
8030 LET L=0
8200 PRINT "  FAIRWAY"
8210 PRINT
8220 PRINT "PRESS ""S"" TO START
, OR ""R"" TO SEE RULES."
8230 IF INKEY$="S" THEN RETURN
8240 IF INKEY$="R" THEN GOTO 830
0
```

```
8250 GOTO 8250

8300 CLS
8310 PRINT " YOU HAVE TO COMPLET
E A 6 HOLE   PUTTING COURSE IN A
S FEW SHOTS"
8320 PRINT "AS POSSIBLE. YOU MUS
T GET THE   BALL - O - , INTO AN
Y PART OF"
8330 PRINT "THE HOLE - ██"; AT 5,
11; "██"
8340 PRINT " YOU CAN HIT THE BAL
L NORTH,     SOUTH, WEST OR EAST,
OR ANY "
8350 PRINT "COMBINATION OF THESE
 AND YOU CAN CHOOSE HOW FAR TO H
IT."
8360 PRINT " YOU ENTER THE INITI
AL OF YOUR  CHOSEN DIRECTION, TH
EN THE"
8370 PRINT "DISTANCE IN METRES.
THE AREA OF  THE SCREEN IS 30*20
 METRES."
8380 PRINT " E.G.- TO GO NORTH 2
0M ,PRESS"
8390 PRINT "  N20, THEN PRESS NE
WLINE. TO GO SOUTH-EAST 9M, PRES
S SE9."
8400 PRINT " GRASS LIKE THIS - ▓
 - IS THICK AND WILL SLOW DOWN T
HE BALL."
8410 PRINT "GRASS LIKE THIS - *
- MAKES THE BALL CHANGE DIRECTIO
N RANDOMLY."
8420 PRINT "( PRESS NEWLINE TO S
TART.)"
8430 INPUT A$
8440 RETURN
```

Alien Descender

Listing occupies 1.5K
Program runs in 2.5K

One lonely alien survived the attack on Earth and landed in the Pacific Ocean. It began to sink down a narrow gorge in the depth of the sea. How long can you, controlling the alien, keep it alive and avoid crashing into the rocky walls of the gorge?

The alien has one weapon which can destroy anything directly below its feet. But this does not protect its 'wings'. Every now and again, trails of bombs are launched from below (inverse asterisks). These can be destroyed by the alien's weapon, or avoided. If they hit the alien, it is destroyed. There are also larger depth charges, shaped like this.

DEPTH CHARGE   BOMB

To destroy a whole depth charge, the alien must be centred exactly above it.

The object is to survive as long as possible.

Use keys 5 and 8 to move left and right respectively and 6 to shoot at the area directly below the alien's feet. At the end, when your alien is destroyed, your score (number of goes survived) is given, along with the current highscore. You are then offered another game.

Remember that as the alien is under water, its reactions are rather sluggish. It moves half a second or so after you press the movement key. If you hold down a key too long, the alien may well blunder on to one wall of the gorge. Naturally, as the alien descends, the gorge gets gradually narrower.

Notes on listing

10–60 Variables. HI = highscore; A = X co-ord of alien; GO = goes; A\$ = space between walls of gorge; RS = co-ord of bomb, if any; X = X co-ord of left side of gorge.

100–150 Print converging walls to straight gorge, so that the alien cannot escape to left or right of gorge.

200 Beginning of main loop. Print left side of gorge, space length A\$ (initially 10 squares, gradually decreases) then right-hand side of gorge.

210 Change X by 1 or not at all. This makes the gorge zig-zag from side to side randomly.

220 Print alien. Note that his arms stretch up when you move.

230–270 PEEK at contents of squares directly under alien and if a bomb, wall or depth charge is there (i.e., not all spaces or newline characters) then goto end of program.

300 Change X co-ord of alien according to key pressed. Note that putting this line here gives the delayed reaction to commands.

310 Increase goes survived.

320 Occasionally decrease A\$, i.e., make the gorge narrower. Note that the minimum size may still let the alien through. Alter this as desired. If LEN A\$ is larger, then it is easier.

330 Occasionally print depth charge on bottom line. As the frequency depends on LEN A\$, more depth charges appear as the gorge gets narrower. Change this line to make depth charges more or less frequent as desired.

400–430 If there is a bomb trail, increase RS randomly and print a new bomb. Start a new bomb trail if RND <.1. (Change this to make bombs more or less frequent, or perhaps dependent on LEN A\$.)

If bombs too far right, stop bomb trail. If bomb trail started, print current bomb. Note that as bombs may be shown to the left of the gorge, some advance warning is given. If you do not wish bombs outside the gorge, change line 430 to

430 IF RS>LEN A\$ THEN etc.

500 Scroll up screen to give idea that alien is moving down-wards.

600 If 6 key pressed, blow up anything directly beneath feet of alien. Note that this does not protect the whole underside of the alien, just the middle squares.

610 Redo main loop.

1000–1210 Alien destroyed, made to flash on and off, score and highscore printed. Screen cleared and new game offered.

Listing

```
   1 REM  ALIEN DESCENDER
   2 REM      COPYRIGHT 1982
              S.ROBERT SPEEL
  10 LET HI=0
  20 LET A=10
  30 LET GO=0
  40 LET A$="            "
  50 LET RS=0
  60 LET X=7
  70 RAND

 100 FOR F=0 TO 7
 110 PRINT AT F,F;"▓▓▓▓";TAB 27-F;
"▓▓▓▓"
 120 NEXT F
 130 FOR F=8 TO 21
 140 PRINT AT F,7;"▓▓▓▓";TAB 19;"▓
▓▓"
 150 NEXT F

 200 PRINT AT 21,X;"▓▓▓▓";A$;"▓▓▓▓"
 210 LET X=X+INT (RND*3)-1-(X=15
)+(X=0)
 220 PRINT AT 0,A;"  ▬▬  ";AT 1,A;
"▐▆▆▌";AT 2,A;" ▆▆ ";AT 3,A;
 230 LET L=PEEK (PEEK 16398+256*
PEEK 16399)
 240 LET L=L+PEEK (1+PEEK 16398+
256*PEEK 16399)
 250 LET L=L+PEEK (2+PEEK 16398+
256*PEEK 16399)
 260 LET L=L+PEEK (3+PEEK 16398+
256*PEEK 16399)
 270 IF L>0 AND L<>236 THEN GOTO
 1000

 300 LET A=A+(INKEY$="8")-(INKEY
$="5")
```

```
 310 LET GO=GO+1
 320 IF RND<.025 AND LEN A$>6 TH
EN LET A$=A$(2 TO )
 330 IF RND<1/LEN A$ THEN PRINT
AT 21,X+RND*LEN A$;"▄▄"

 400 IF RS>0 THEN LET RS=RS+RND*
7
 410 LET RS=RS+(RND<.1)
 420 IF RS>LEN A$+X THEN LET RS=
0
 430 IF RS>0 THEN PRINT AT 21,RS
;"▓"
 500 SCROLL

 600 IF INKEY$="6" THEN PRINT AT
3,A+1;"==";AT 3,A+1;"▄▄";AT 3,A
+1;"
 610 GOTO 200

1000 FOR F=1 TO 10
1010 PRINT AT 0,A;" ▄▄ ";AT 1,A;
"▐██▌";AT 2,A;" ▄▄ "
1020 PRINT AT 0,A;"    ";AT 1,A;"
    ";AT 2,A;"    "
1030 NEXT F

1050 PRINT AT 20,0;"YOUR SCORE I
S ";GO
1060 IF GO>HI THEN LET HI=GO
1070 PRINT AT 21,2;"HISCORE = ";
HI
1080 FOR F=1 TO 200
1090 NEXT F
1100 CLS
1110 PRINT "ANOTHER GAME? (Y/N)"
1120 IF INKEY$="Y" THEN GOTO 20
1130 IF INKEY$="N" THEN STOP
1210 GOTO 1120
```

Alien Descender Adaptation for Spectrum
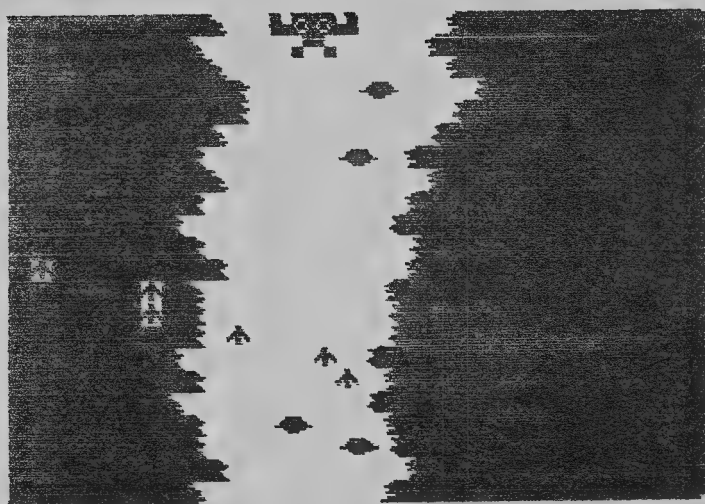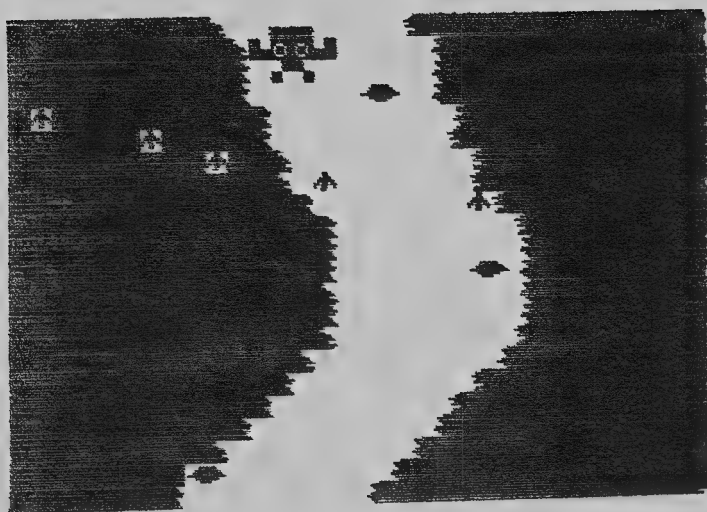
This program needs a lot of re-programming for the Spectrum, as in addition to adding colour and some sound, the object detection section, which checks if you have lost, must be changed. Also, some more interesting shapes for the bombs and depth charges will be user-defined.

Bomb

Depth Charge

Notes on altered listing for Spectrum

10 Hi is highscore, the picture will be sea blue (cyan).

20 Set up user-defined characters for bomb, depth charge and sides of cavern.

110, 140 and 200 The sides of the cavern are dark blue, and have a choice of graphics.

Note that the spaces between the walls have a black ink colour even though no ink colour may be seen, as ATTR is used in the program. The rest of the screen, outside the walls, is made dark, which is impracticable on the ZX81 due to its slower speed of operation.

240–260 We cannot PEEK on screen easily on the Spectrum and SCREEN\$ will not recognize the existence of user-defined characters, so ATTR is used to check if you have hit anything. It considers anything which does not have black ink as a crash. Note that if you wish to change the colours, brightness, etc., you must change line 260 to 4 times the ATTR value of a blank square.

330 Depth charge.

430 Bomb.

500 Spectrum equivalent of SCROLL.

610 Short 'movement' sound.

620–650 Firing sequence. 'Laser' sound – note same length as 'movement' sound so that the game will not slow down each time you fire.

1000–1020 Score and highscore shown prominently.

1040–1050 Exploding alien and random sound.
1200–1260 Data. 1200 = bomb. 1210–1220 = depth charge. 1230–1270 = sides of cavern.

Listing

```
   1 REM Alien Descender
   2 REM  © S.Robert Speel 1982

  10 LET hi=0: BORDER 5: PAPER 5
CLS
  20 FOR f=1 TO 7: FOR g=0 TO 7:
READ b: POKE USR CHR$ (143+f)+g
,b: NEXT g: NEXT f
  30 LET go=0
  40 LET a$=""
  50 LET r$=0
  60 LET x$="                "
  70 LET y$=x$+x$: LET y$=y$+y$
  80 LET a=10
  90 RANDOMIZE

 100 FOR f=0 TO 7
 110 PRINT INK 1;AT f,0;y$( TO f
);CHR$ (147+INT (RND*2)); INK 0;
TAB 27-f; INK 1;CHR$ (149+INT (R
ND*2));y$( TO 30-LEN x$-f)
 120 NEXT f
 130 FOR f=8 TO 21
 140 PRINT INK 1;AT f,0;y$( TO 7
);CHR$ (147+INT (RND*2)); INK 0;
TAB 19; INK 1;CHR$ (149+INT (RND
*2));y$( TO 12)
 150 NEXT f

 200 PRINT INK 1;AT 21,0;x$;CHR$
 (147+INT (RND*2)); INK 0,a$; IN
K 1;CHR$ (149+INT (RND*2));y$( T
O 30-LEN (x$+a$))
 210 LET x$=x$(2 TO )+y$( TO INT
 (RND*3)): IF LEN x$>14 THEN LET
 x$=x$(2 TO )
 220 IF LEN x$<1 THEN LET x$=x$+
".."
 230 PRINT AT 0,a;"  ___  ";AT 1,a;
"       ";AT 2,a;"  ___  "
 240 LET l=0
 250 FOR f=0 TO 3: LET l=l+ATTR
(3,a+f): NEXT f
 260 IF l>160 THEN GO TO 1000
```

```
  300 LET a=a+(INKEY$="8")-(INKEY
$="5")
  310 LET go=go+1
  320 IF RND<.025 AND LEN a$>6 TH
EN LET a$=a$(2 TO )
  330 IF RND<1/LEN a$ THEN PRINT
INK 3;AT 21,LEN x$+RND*LEN a$;CH
R$ 145;CHR$ 146

  400 IF rs>0 THEN LET rs=rs+RND*
7
  410 LET rs=rs+(RND<.1)
  420 IF rs>LEN a$+LEN x$ THEN LE
T rs=0
  430 IF rs>0 THEN PRINT INK 2;AT
21,rs;CHR$ 144

  500 PRINT AT 21,0: POKE 23692,2
55: PRINT

  600 IF INKEY$="6" THEN GO TO 62
0
  610 BEEP .04,0: GO TO 200
  620 PRINT AT 3,a+1;"==";AT 3,a+
1;"    "
  630 BEEP .015,30: BEEP .02,22:
BEEP .02,14
  640 PRINT AT 3,a+1;"    "
  650 GO TO 200

1000 PRINT PAPER 6;AT 20,0;"Your
 score is ";go
1010 IF go>hi THEN LET hi=go
1020 PRINT FLASH 1;AT 0,a;"    "
;AT 1,a;"     ";AT 2,a;"     "
1030 PRINT PAPER 6;AT 21,1;" hi-
score = ";hi
1040 FOR f=1 TO 100: BEEP .01,RN
D*50-RND*50: NEXT f
1050 PRINT FLASH 0;AT 0,a;"    "
;AT 1,a;"     ";AT 2,a;"     "
1060 FOR f=1 TO 600: NEXT f

1100 CLS
1110 PRINT "Another game? (y/n)"
1120 IF INKEY$="y" THEN GO TO 30
1130 IF INKEY$="n" THEN STOP
1140 GO TO 1120

1200 DATA 24,24,60,126,219,153,2
4,0
1210 DATA 0,7,15,31,127,15,3,0
1220 DATA 0,224,240,248,254,240,
192,0
```

```
1230 DATA 248,252,255,254,248,25
2,254,255
1240 DATA 224,192,240,252,254,24
8,255,254
1250 DATA 1,15,7,3,15,31,7,3
1260 DATA 7,31,127,31,63,7,31,63
```

Asteroid Belt

Listing occupies 2.2K
Program runs in 3.3K

To use

You have a small spaceship which has blundered into the
asteroid belt. Most of the ship's functions have been disabled
by a collision with a small meteor, leaving you with only the
commands left and right. How long can you dodge the asteroids
hurtling up at you before you are hit?

You control your spaceship's movement with the 5 and 8
keys. In the ZX81 version the asteroids appear one at a time
when you start, but as you go deeper into the belt, they come
in twos and threes or even four at a time.

A highscore feature is included, giving the top three scores
so far.
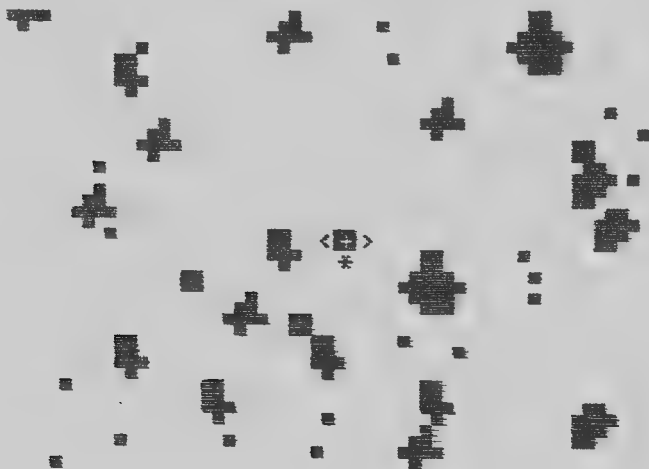
Notes on listing

10 Goto instructions and variables.

20–190 Different asteroids. Note that often asteroids link up to form small planetoids.

500–530 Choose asteroid to PRINT AT bottom of screen. Line 500 allows for increase after 50, 100 and 150 goes up to four asteroids per go.

600–630 Increment go counter, move spaceship according to INKEY\$, SCROLL the asteroids up the screen, and PRINT spaceship.

700–740 Check if ship has crashed by considering three print-co-ordinates under ship. IF OK, goto 500.

2000–2030 You crash. Explosion shown.

2100–2260 Your score and highscore announced and updated. New game offered.

5000–5050 Variables reset at start of each game, screen cleared and spaceship drawn for new game.

7000–7090 Short rules. These can be expanded if you wish.

7100–7160 Start first game. H1, H2, and H3 set to 0 – these are the three best scores so far. Clear screen and goto 5000.

Listing

```
 1 REM    ASTEROID BELT
 2 REM    COPYRIGHT 1982
           S.ROBERT SPEEL

10 GOTO 7000

20 PRINT AT 21,A;"█ "
30 RETURN
40 PRINT AT 21,A;" █"
50 RETURN
60 PRINT AT 21,A;"▄ "
70 RETURN
80 PRINT AT 21,A;" ▄"
90 RETURN
100 PRINT AT 21,A;"█"
110 RETURN
120 PRINT AT 20,A;"█";AT 21,A;"
▀ "
130 RETURN
140 PRINT AT 20,A;" ▄ ";AT 21,A;
"▄█▄"
150 RETURN
```

```
 160 PRINT AT 20,A;"█▄";AT 21,A;
"██"
 170 RETURN
 180 PRINT AT 19,A+1;"█";AT 20,A
;"███";AT 21,A;"██"
 190 RETURN

 500 FOR F=1 TO 1+(GO>50)+(GO>10
0)+(GO>150)
 510 LET A=INT (RND*30)
 520 GOSUB 20+INT (RND*9)*20
 530 IF RND<.8 THEN NEXT F

 600 LET GO=GO+1
 610 LET X=X+(INKEY$="8")-(INKEY
$="5")+(X<1)-(X>28)
 620 SCROLL
 630 PRINT AT 9,X-1;"      ";AT 1
0,X;"<█>";AT 11,X;" *";AT 12,X;

 700 LET Q=PEEK (PEEK 16398+256*
PEEK (16399))
 710 LET R=PEEK (1+PEEK 16398+25
6*PEEK (16399))
 720 LET S=PEEK (2+PEEK 16398+25
6*PEEK (16399))
 730 IF Q+R+S>118 AND Q+R+S<234
THEN GOTO 2000
 740 GOTO 500

2000 FOR J=1 TO 10
2010 PRINT AT 10,X;" ▓ ";AT 11,X
;"   ";AT 10,X;"▓▓▓▓";AT 9,X;"▓▓▓▓
";AT 11,X;"▓▓▓"
2020 PRINT AT 10,X+1;" ";AT 9,X;
"    ";AT 10,X;"    ";AT 11,X;"
"
2030 NEXT J

2100 PRINT AT 5,5;"YOU CRASHED";
AT 6,5;"YOUR SCORE IS ";GO
2110 IF GO>H2 THEN LET H3=H2
2120 IF GO>H1 THEN LET H2=H1
2130 IF GO>H1 THEN LET H1=GO
2140 IF GO<H1 AND GO>H2 THEN LET
 H2=GO
2150 IF GO<H2 AND GO>H3 THEN LET
 H3=GO

2200 PRINT AT 8,5;"HIGHEST SCORE
 SO FAR = ";H1
2210 PRINT AT 9,5;"SECOND HIGHES
T = ";H2
```

```
2220 PRINT AT 10,5;"THIRD HIGHES
T = ";H3
2230 PRINT AT 12,3;"ANOTHER GO?
(Y/N)"
2240 IF INKEY$="Y" THEN GOTO 500
0
2250 IF INKEY$="N" THEN STOP
2260 GOTO 2240

5000 RAND
5010 LET X=15
5020 LET GO=0
5030 CLS
5040 PRINT AT 10,X;"<■>";AT 11,X
;" ‡"
5050 GOTO 500

7000 PRINT "     ASTEROID BELT"
7010 PRINT
7020 PRINT " IN THIS GAME YOU HA
VE TO AVOID"
7030 PRINT "THE ASTEROIDS COMING
 TOWARDS YOU"
7040 PRINT "BY MOVING YOUR SPACE
SHIP WITH "
7050 PRINT "THE ""5"" AND ""8""
KEYS."
7060 PRINT "YOUR SHIP LOOKS LIKE
 THIS:-"
7070 PRINT AT 8,10;"<■>";AT 9,11
;" ‡"
7080 PRINT "TO MOVE, HOLD DOWN T
HE 5 OR 8 "
7090 PRINT "KEY CONTINUOUSLY."

7100 PRINT " PRESS NEWLINE TO ST
ART."
7110 INPUT A$
7120 CLS
7130 LET H1=0
7140 LET H2=0
7150 LET H3=0
7160 GOTO 5000
```

**Asteroid Belt**   Conversion for Spectrum

Due to the difference in PLOTting on the Spectrum, the ZX81 listing has been extensively remodelled to make use of the Spectrum's features.

There are several differences noticeable while playing. The asteroids are lightly coloured on a black background, and initially only small ones appear. After a while, larger 'moonlets' appear, and then the asteroids appear in larger numbers. If you still survive, darker, easily missed asteroids appear, and flashing ones help distract you from the less conspicuous rocks.

You move left and right using the 5 and 8 keys, and the top three scores (and your score) are given at the end of the game when you crash.
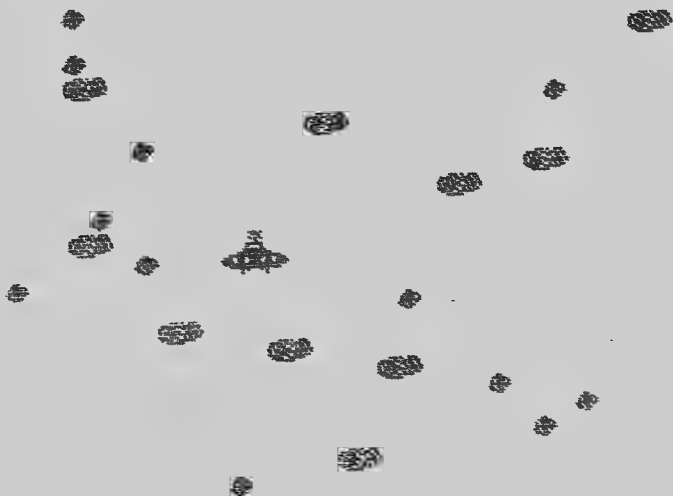
Notes on listing

10   Goto rules, variables, etc.

20–40   Three different sizes of asteroids, PRINTed with user-defined graphics.

500   PRINT 1 or 2 asteroids, depending on how long you have survived, and a random factor.

600–610 Increment go counter, move spaceship according to key pressed. Check if you hit asteroid.

620 SCROLL.

630–640 PRINT flashing spaceship and repeat main loop.

2000–2010 Your spaceship explodes. 'Blowing up' effect done with user-defined 'explosion' graphic being PRINTed on spaceship in darker and darker colours until it disappears.

2100–2240 Your score and top three scores updated and given. Another game offered. Note that if refused (line 2230), INK, PAPER and BORDER colours changed to normal, so you can see listing. If game accepted (line 2220) you do not see rules again, and highscore variables not reset.

5000–5010 Reset go counter, and position spaceship for new game.

6000 Set up user-defined spaceship, explosion and asteroids.

7000–7030 Rules. Only shown at beginning of first game. You can add a 'see rules' option at the beginning of a new game, but you will have to move line 7050 (except the last statement) to somewhere else, e.g., 6500.

7040 Sets the screen to a black outer-space view.

7050 h1, h2 and h3 are the highscores and *not* reset after each game.

9000–9050 Data for user-defined graphics used to make spaceship, asteroids and explosion.

Listing

```
   1 REM   Asteroid Belt
   2 REM   © S.Robert Speel 1982

  10 GO TO 6000

  20 PRINT FLASH ((go)125) AND R
ND<.5); BRIGHT 1;AT 21,a;CHR$ 14
8: RETURN
  30 PRINT BRIGHT 1;AT 21,a;CHR$
149;CHR$ 150: RETURN
  40 PRINT INK 4; BRIGHT 1;AT 20
,a;CHR$ 151; INK 6;CHR$ 153;AT 2
1,a;CHR$ 152; INK 4;CHR$ 154: RE
TURN
```

```
 500 INK 4-(RND<.5 AND go>100)*3
: OVER 1: FOR f=1 TO 1+(go>100):
 LET a=INT (RND*30): GO SUB 20+I
NT (RND*(2+(go>50)))*10: IF RND<
.8 THEN NEXT f

 600 OVER 0: LET go=go+1: LET x=
x+(INKEY$="8")-(INKEY$="5")+(x<1
)-(x>28): BEEP .005,0: BEEP .004
,40
 610 IF ATTR (12,x)>10 OR ATTR (
12,x+1)>10 OR ATTR (12,x+2)>10 T
HEN GO TO 2000
 620 POKE 23692,255: PRINT AT 21
,0''
 630 PRINT AT 9,x;"      "; INK 5;
AT 10,x-1;"   "; FLASH 1;CHR$ 144
; FLASH 0;"   "; FLASH 1; INK 2;A
T 11,x;CHR$ 146; FLASH 0;CHR$ 14
5; FLASH 1;CHR$ 147
 640 GO TO 500


2000 FOR f=7 TO 0 STEP -1: BEEP
.1,f
2010 INK f: PRINT AT 11,x+1;CHR$
 155: BEEP .01,-20: FOR g=1 TO 3
: PRINT AT 9+g,x;CHR$ 155;CHR$ 1
55;CHR$ 155;CHR$ 155: NEXT g: NE
XT f
2100 INK 7: PAPER 2: PRINT AT 5,
5;"You Crashed"'"Your score is "
;go'
2110 IF go>h2 THEN LET h3=h2
2120 IF go>h1 THEN LET h2=h1: LE
T h1=go
2130 IF go<h1 AND go>h2 THEN LET
 h2=go
2140 IF go<h2 AND go>h3 THEN LET
 h3=go

2200 PRINT AT 8,5;"Highest score
 so far = ";h1;TAB 5;"2nd Highes
t = ";h2;TAB 5;"3rd Highest = ";
h3
2210 PRINT AT 12,3;"Another go?
(y/n) "
2220 IF INKEY$="y" THEN GO TO 50
00
2230 IF INKEY$="n" THEN PAPER 7:
 BORDER 7: INK 0: STOP
2240 GO TO 2220
```

```
5000 RANDOMIZE : LET x=15: LET g
o=0: PAPER 0: CLS
5010 PRINT AT 10,x+1;CHR$ 144;AT
 11,x;CHR$ 146;CHR$ 145;CHR$ 147
5020 GO TO 500

6000 FOR f=0 TO 11: FOR g=0 TO 7
: READ a: POKE USR CHR$ (144+f)+
g,a: NEXT g: NEXT f

7000 PRINT "    Asteroid Belt"'''"
In this game you have to avoid t
he asteroids coming towards you"
7010 PRINT "by moving your space
ship with  the "5" & "8" keys."
7020 PRINT "Your spaceship looks
 like this:-";AT 10,15;CHR$ 144;
AT 11,14;CHR$ 145;CHR$ 145;CHR$
147
7030 PRINT '"To move,hold down t
he 5 or 8 key continuously."'"Pr
ess ENTER to start"
7040 INPUT a$: PAPER 0: BORDER 0
: INK 0: CLS
7050 LET h1=0: LET h2=0: LET h3=
0: GO TO 5000
9000 DATA 0,42,28,42,8,126,66,25
5,255,255,60,60,255,255,66,0
```

```
9010 DATA 3,31,115,243,127,31,1,
1,192,248,206,207,254,248,128,12
8
9020 DATA 24,62,93,127,218,108,2
4,0,14,123,220,183,189,238,121,3
1,156,118,218,45,253,86,248,128

9030 DATA 0,5,31,53,105,121,95,2
54,255,215,54,27,30,15,1,0
9040 DATA 0,128,224,184,204,115,
106,215,253,61,166,124,244,116,2
16,176
9050 DATA 16,130,40,17,40,130,16
,66
```
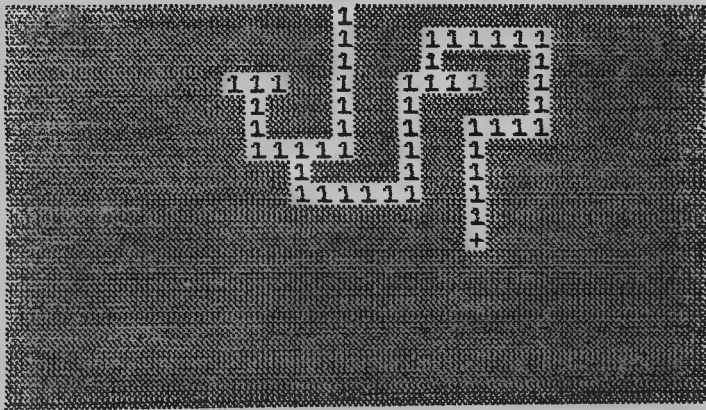
Amaze

Listing occupies 1.8K
Program runs in 2.8K

To play
It is degrading for a computer merely to draw mazes for us to
solve like experimental animals. In this game you draw a maze
and the computer solves it.

On RUNning the program you are presented with a grey
block, filling most of the screen. There is a blinking '1' in the
top centre and this is your cursor. You move using keys 5, 6,
7 and 8. As you draw the maze a trail of 1s appears. You exit
the maze anywhere along the bottom line and just hold the 6
key down until the computer's 'finger' appears. The
computer's 'finger' is an inverse flashing + sign. The computer
solves the maze by attempting to go downwards. If it can't, it
tries first left then right and then upwards. Each time it passes
over a number it increases it by one. It always tries to go on to
the lowest number, and this bias precedes all others. This is not
the fastest way of solving a maze but it is certainly one of the
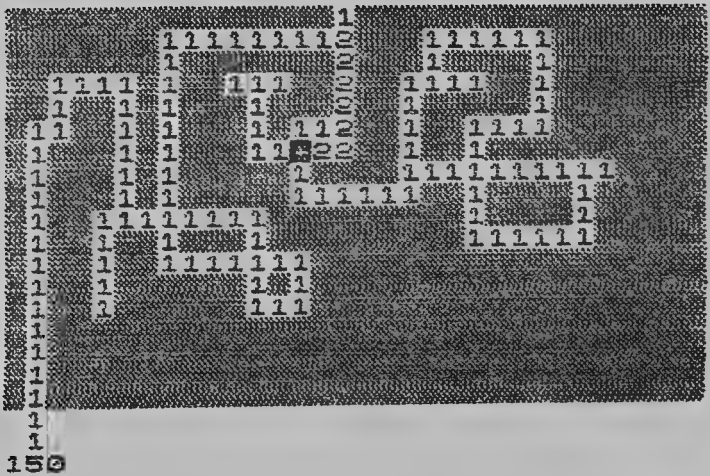most interesting and difficult to predict.

When your computer finishes the maze it gives your score
according to how long it took to solve your maze, as well as the
current highscore.

You must create your maze within about 150 goes and your
number of goes taken is shown to the nearest ten below, in the
bottom left of the screen. You may remove this restriction and
go for an ultimate highscore.

A maze being drawn.

The maze is finished and the computer has started solving it.

Notes on listing

30 Goto start of game and variables.

40 D contains contents of squares around computer's finger.

50–190 Computer PRINTs at each location around its finger and then PEEKs at what is in it. These values are stored in D(1) to D(4). D(5) contains the contents of the square which the finger covers and D(6) the contents of the square two to the right of the finger, to counter the computer's left bias. It might seem simpler to have all these lines in a FOR-NEXT loop, with the PRINT locations in a string array, but this works more slowly.

400 Print number which was under 'finger' + 1 over finger.

410–440 Make sure contents of D are within acceptable limits, if any of the surrounding squares contain '1' goto sub-routine to move that way.

450 If next square but one to right contains a 1, and there is a way through, goto 'go right' sub-routine. This partly counterbalances the computer's preference of left to right.

600–640 Find lowest number surrounding a move in that direction. This has been separated from the loop in lines 400–440 to speed up movement when there are lots of 1s about.

1000 Move down.

1020–1030 General movement sub-routine. Increment 'go counter', check if end of maze reached. If so, goto 2000. Redo main loop.

1100 Move left.

1200 Move right.

1300 Move up.

Note that this order gives the computer its downward bias, so that it will always seek the exit at the bottom, other things being equal.

2000–2080 End of game. Print how long the computer took and highscore at present. Offers another go, which if accepted, re-runs the program, but without resetting RS, the highscore.

9000–9060 Variables. RS = highscore, Y and X are initial position of cursor, A = initial number of moves you make in drawing the maze. Also maze framework drawn.

Listing

```
   10 REM AMAZE
   20 REM      COPYRIGHT 1982
            S. ROBERT SPEEL.
   30 GOTO 9000
   40 DIM D(6)
   50 PRINT AT Y,X;
   60 LET D(5)=PEEK (PEEK 16398+2
56*PEEK 16399)
   70 PRINT AT Y,X;"█"

  100 PRINT AT Y+1,X;
  110 LET D(1)=PEEK (PEEK 16398+2
56*PEEK 16399)
  120 PRINT AT Y,X-1;
  130 LET D(2)=PEEK (PEEK 16398+2
56*PEEK 16399)
  140 PRINT AT Y,X+1;
  150 LET D(3)=PEEK (PEEK 16398+2
56*PEEK 16399)
  160 PRINT AT Y-1,X;
  170 LET D(4)=PEEK (PEEK 16398+2
56*PEEK 16399)
  180 PRINT AT Y,X+2;
  190 LET D(6)=PEEK (PEEK 16398+2
56*PEEK 16399)

  400 PRINT AT Y,X;CHR$ (D(5)+1)
  410 FOR F=1 TO 6
  420 IF D(F)<29 OR D(F)>37 THEN
LET D(F)=37
  430 IF F<5 AND D(F)=29 THEN GOT
O 900+100*F
  440 NEXT F
  450 IF D(6)=29 AND D(3)<37 THEN
GOTO 1200
  600 FOR F=30 TO 36
  610 FOR G=1 TO 4
  620 IF D(G)=F THEN GOTO 900+G*1
00
  630 NEXT G
  640 NEXT F

 1000 LET Y=Y+1
 1010 LET GO=GO+1
 1020 IF Y=18 THEN GOTO 2000
 1030 GOTO 50

 1100 LET X=X-1
 1110 GOTO 1010
```

Amaze

Listing occupies 1.8K
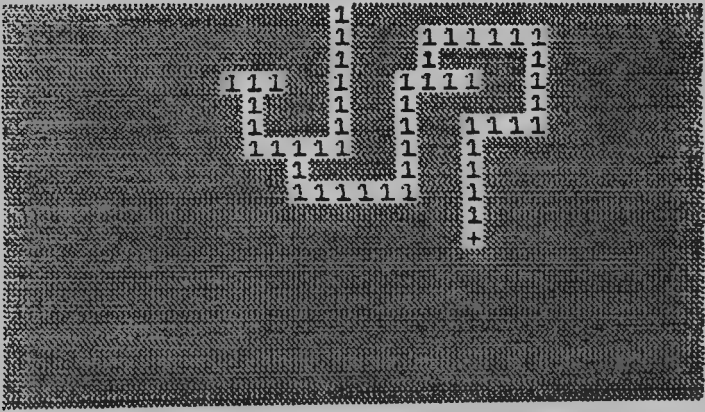Program runs in 2.8K

To play
It is degrading for a computer merely to draw mazes for us to
solve like experimental animals. In this game you draw a maze
and the computer solves it.

On RUNning the program you are presented with a grey
block, filling most of the screen. There is a blinking '1' in the
top centre and this is your cursor. You move using keys 5, 6,
7 and 8. As you draw the maze a trail of 1s appears. You exit
the maze anywhere along the bottom line and just hold the 6
key down until the computer's 'finger' appears. The
computer's 'finger' is an inverse flashing + sign. The computer
solves the maze by attempting to go downwards. If it can't, it
tries first left then right and then upwards. Each time it passes
over a number it increases it by one. It always tries to go on to
the lowest number, and this bias precedes all others. This is not
the fastest way of solving a maze but it is certainly one of the
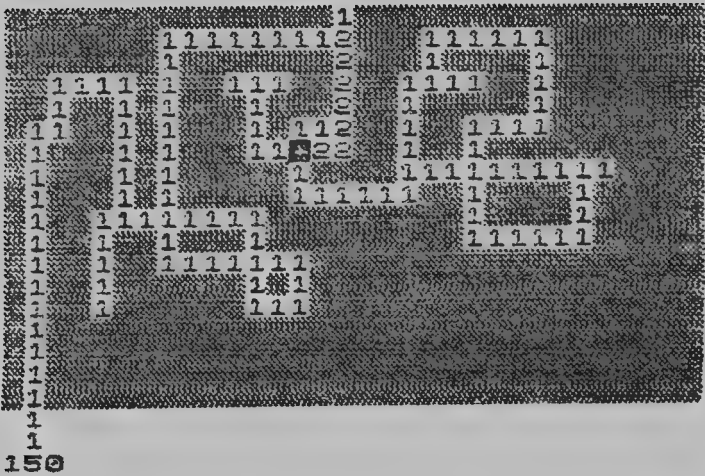most interesting and difficult to predict.

When your computer finishes the maze it gives your score
according to how long it took to solve your maze, as well as the
current highscore.

You must create your maze within about 150 goes and your
number of goes taken is shown to the nearest ten below, in the
bottom left of the screen. You may remove this restriction and
go for an ultimate highscore.

50

A maze being drawn.

150

The maze is finished and the computer has started solving it.

Amaze
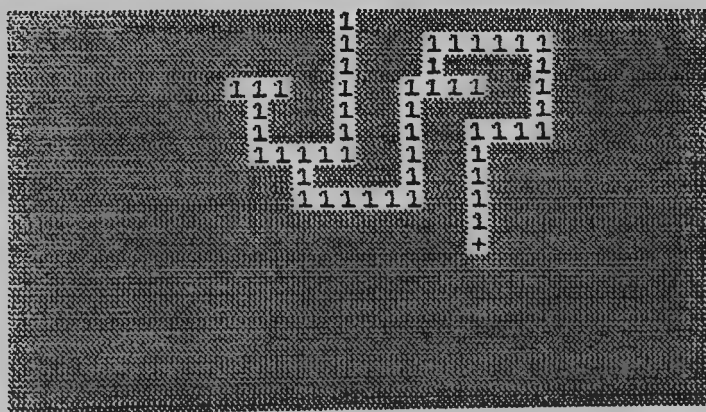
Listing occupies 1.8K
Program runs in 2.8K

To play
It is degrading for a computer merely to draw mazes for us to solve like experimental animals. In this game you draw a maze and the computer solves it.

On RUNning the program you are presented with a grey block, filling most of the screen. There is a blinking '1' in the top centre and this is your cursor. You move using keys 5, 6, 7 and 8. As you draw the maze a trail of 1s appears. You exit the maze anywhere along the bottom line and just hold the 6 key down until the computer's 'finger' appears. The computer's 'finger' is an inverse flashing + sign. The computer solves the maze by attempting to go downwards. If it can't, it tries first left then right and then upwards. Each time it passes over a number it increases it by one. It always tries to go on to the lowest number, and this bias precedes all others. This is not the fastest way of solving a maze but it is certainly one of the most interesting and difficult to predict.
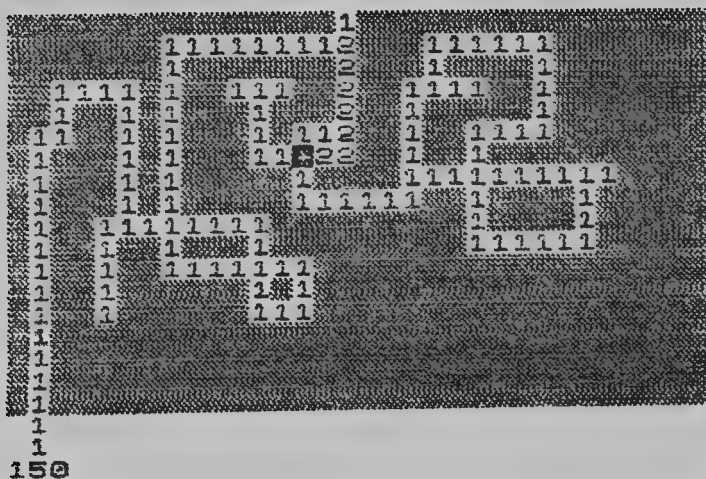
When your computer finishes the maze it gives your score according to how long it took to solve your maze, as well as the current highscore.

You must create your maze within about 150 goes and your number of goes taken is shown to the nearest ten below, in the bottom left of the screen. You may remove this restriction and go for an ultimate highscore.

50

A maze being drawn.

150

The maze is finished and the computer has started solving it.

```
1200 LET X=X+1
1210 GOTO 1010

1300 LET Y=Y-1
1310 GOTO 1010

2000 PRINT AT Y,X;"■"
2010 PRINT AT 19,0;"FINISHED IN
";GO;" GOES."
2020 IF GO>RS THEN LET RS=GO
2030 PRINT "BEST SO FAR = ";RS
2040 PRINT "ANOTHER GO? (Y/N)"
2050 IF INKEY$<="" THEN GOTO 205
0
2060 IF INKEY$="N" THEN STOP
2070 CLS
2080 GOTO 9010

9000 LET RS=0
9010 FOR F=1 TO 18
9020 PRINT "███████████████████████████"
9030 NEXT F
9040 LET Y=0
9050 LET X=15
9060 LET A=0

9100 PRINT AT Y,X;"→"
9110 PRINT AT Y,X;"1"
9120 LET Z=X+Y
9130 LET X=X+(INKEY$="8")-(INKEY
$="5")-(X>29)+(X<2)
9140 LET Y=Y+(INKEY$="6")-(INKEY
$="7")+(Y<0)
9150 IF Y>18 THEN GOTO 9200
9160 IF Z<>X+Y THEN LET A=A+1
9170 IF A>150 THEN GOTO 9300
9180 IF A/10=INT (A/10) THEN PRI
NT AT 20,0;A
9190 GOTO 9100

9200 LET X=15
9210 LET Y=1
9220 LET GO=0
9230 GOTO 40

9300 FOR F=Y TO 19
9310 PRINT AT F,X;"1"
9320 NEXT F
9330 GOTO 9200
```

Vapours on Venus

Listing occupies 1.8K
Program runs in 2.8K

To play

You are in command of a spaceship orbiting Venus. A robot probe has been sent down to get data. However, a highly corrosive black gas is spreading on the surface and if it touches the robot, will destroy it. There is also a harmless grey gas, which may push back the black vapours.

By using keys 5, 6, 7 and 8 to move the robot probe, you must keep it out of the black gas as long as possible. Sometimes the grey gas forms a safe path through the black gas.

As the probe moves, it leaves a trail of asterisks as beacons, and these are what you see on screen. The beacon at which the probe is stationed flashes.

A score of more than 50 seconds is quite good.

```
    *
     *
     *
     *
     *
     * * * * * *
         *
         *
         *
         *


         * *
```
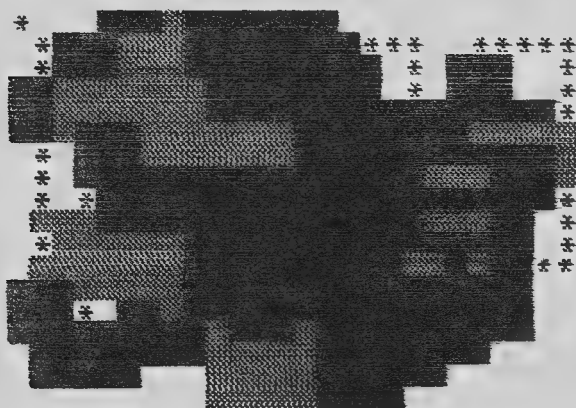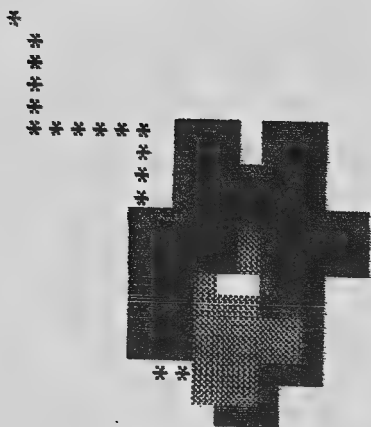
Notes on listing

100–340 'Gas cloud' loop. There are 4 clouds, 3 black, 1 grey.

110–120 X(F) and Y(F) are the PRINT co-ords for each cloud. These change partly at random, but tend to follow the robot probe. To increase the difficulty, change the (RND <.5) parts of lines 110–120 to (RND <.2) or even (RND <.1). This will make the clouds converge on the probe faster and more purposefully. Note that clouds cannot go off-screen.

200–210 Print clouds, 3 black, 1 grey.

220–230 Change robot probe's position according to key pressed. Note that the robot cannot go further than the clouds in any direction. Note also that you move four times as fast as any one cloud.

300–320 Check to see if probe in black cloud. If it is, goto end of game routine.

330 Increment score in seconds. The real time per go is very nearly one second, sixty game seconds are approximately sixty-five real seconds.

340–350 Repeat for all four clouds and then do again.

1000–1140 End of game routine.

1000–1020 Destroyed robot.

1060–1080 Check to see if new highscore, print highscore if

suitably large highscore obtained, print congratulations.
1100–1140 Offers another game; if accepted, resets all
variables except HI, the current highscore.

8000–8120 Variables.

HI = highscore, X(F), Y(F) are PRINT co-ordinates of
centres of 'gas currents'. Note that all currents start fom the
centre of the screen.

9000–9100 Rules. Expand or omit as desired.

Listing

```
   1 REM  VAPOURS ON VENUS
   2 REM        COPYRIGHT 1982
               S.ROBERT SPEEL
  10 GOTO 8000
 100 FOR F=1 TO 4
 110 LET X(F)=X(F)+(1-2*(RND<.5)
)-(X(F)>28)+(X(F)<2)+(X(F)<X)-(X
(F)>X)
 120 LET Y(F)=Y(F)+(1-2*(RND<.5)
)+(Y(F)<Y)-(Y(F)>18)+(Y(F)<2)-(Y
(F)>Y)

 200 IF F=1 THEN PRINT AT Y(F),X
(F);"▓▓▓▓";AT Y(F)+1,X(F);"▓▓▓▓";A
T Y(F)+2,X(F);"▓▓▓▓"
 204 IF F>1 THEN PRINT AT Y(F),X
(F);"▓▓▓";AT Y(F)+1,X(F);"▓▓▓";A
T Y(F)+2,X(F);"▓▓▓"
 210 IF F>1 THEN PRINT AT Y(F),X
(F);"▓▓▓";AT Y(F)+1,X(F);"▓▓▓";A
T Y(F)+2,X(F);"▓▓▓"
 220 LET X=X+(INKEY$="8")-(INKEY
$="5")+(X<2)-(X>29)
 230 LET Y=Y+(INKEY$="6")-(INKEY
$="7")+(Y<2)-(Y>17)

 300 PRINT AT Y,X;
 310 IF PEEK (PEEK 16398+256*PEE
K 16399)=128 THEN GOTO 1000
 320 PRINT AT Y,X;" ";AT Y,X;"*"
 330 LET SC=SC+1
 340 NEXT F
 350 GOTO 100

1000 FOR F=1 TO 10
1010 PRINT AT Y,X;"▓";AT Y,X;"*"
1020 NEXT F
1030 CLS
```

```
1040 PRINT "YOU SURVIVED FOR ";S
C;" SECS."
1050 PRINT
1060 IF SC>HI THEN LET HI=SC
1070 PRINT "HIGH SCORE = ";HI
1080 IF SC=HI AND SC>50 THEN PRI
NT "CONGRATULATIONS."
1090 PRINT
1100 PRINT "ANOTHER GAME?(Y/N) "
1110 IF INKEY$="" THEN GOTO 1110
1120 IF INKEY$="N" THEN STOP
1130 CLS
1140 GOTO 8010

8000 LET HI=0
8010 DIM X(4)
8020 DIM Y(4)
8030 FOR F=1 TO 4
8040 LET X(F)=15
8050 LET Y(F)=10
8060 NEXT F

8100 LET Y=1
8110 LET X=1
8120 LET SC=0
8130 RAND

9000 PRINT " VAPOURS ON VENUS"
9010 PRINT
9020 PRINT "YOU HAVE SENT A ROBO
T PROBE TO  VENUS. THE PROBE CAN
 BE ENGULFED BY THE SPREADING BL
ACK CLOUDS."
9030 PRINT "THERE ARE ALSO GREY,
 HARMLESS    CLOUDS OF VAPOUR, AN
D THESE MAY PUSH ASIDE THE BLACK
 GASES."
9040 PRINT " HOW LONG CAN YOU KE
EP THE PROBE TRANSMITTING BEFORE
 THE BLACK  CLOUDS DESTROY IT?"
9050 PRINT " TO MOVE THE ROBOT,
USE THE KEYS 5,6,7 AND 8 ON YOUR
 COMPUTER AT MISSION CONTROL, TH
E ZX8000001."
9060 PRINT

9070 PRINT " (PRESS NEWINE TO ST
ART)"
9080 IF INKEY$="" THEN GOTO 9080
9090 CLS
9100 GOTO 100
```

Rabbits

Listing occupies 3.8K
Program runs in 4.9K

To play

This is a genetics game. You are a rabbit breeder and must make as much profit as you can within five years. You start with eight rabbits, out of which you keep two and sell the rest. These two produce eight offspring for the next year, and again you choose two. After five years, your profit is announced, along with the current highscore.

The profit you make on a rabbit depends on its coat colour and type of ears. A patchy rabbit with an unsymmetrical pattern is suitable only as a pet, and is worth only a few pounds. A rabbit with a mono-coloured coat is worth about £10. Most rabbits have medium-sized ears, so rabbits with slim ears or wide ears are worth a little more than common ones.

On running the program, you are presented with a shortened version of the rules. Press NEWLINE and after a short pause the first generation male rabbits will appear. You may select one out of the four by pressing key 1, 2, 3 or 4 as appropriate. The value of each rabbit is given. You then select a female rabbit from the four females which will be shown on the screen after you have selected the male. These will breed the next generation.
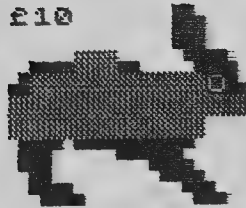
Many colours are borne by recessive genes, such as 'spotted'. 'Black' is a dominant strain, so a rabbit with one 'black' and one 'spotted' colour gene will be black. However, some genes are co-dominant, such as 'black+grey' and thus a rabbit with 'black' and 'grey' colour genes may be black or grey, or striped grey and black.
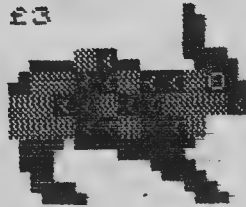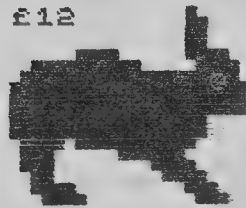
£12 1
£10 3
£15 2
£3 4

MALES, GENERATION 1
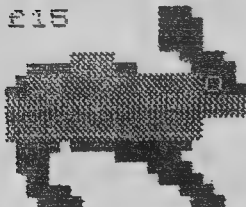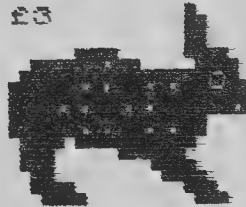CHOOSE THE LUCKY RABBIT

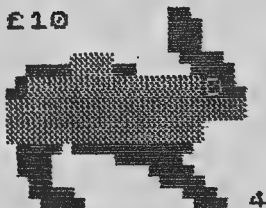£12 1
£12 3
£15 2
£3 4

FEMALES, GENERATION 1
CHOOSE THE LUCKY RABBIT

FEMALES, GENERATION 2
CHOOSE THE LUCKY RABBIT

Occasionally, a rarer colour such as 'X patterned' can occur and these are worth a lot of money. Often two colours will combine to provide a blotchy effect which is practically worthless.

The ears are simpler, and can be wide, medium or narrow, narrow being the most dominant. However, a grey rabbit may occasionally have large grey ears, and a spotted rabbit may have spotted ears, both of which add to the rabbit's value. Very occasionally, a rabbit throwback may have 'asterisked' ears, but these, as well as other coloured ears, are not usually inherited by the offspring.

Notes on listing
10 Gosub starting variables and rules.
20 Z = 1 for males, 2 for females.
30 CT = rabbit number.
40–50 Beginning of loop to draw four rabbits.
70 Go and find rabbit colour and ear-type.
80 Print value of rabbit.
100–120 Draw rabbit, with correct-shaped ears. Note that *in these lines only* black squares (inverse spaces) have been replaced with inverse full stops for ease of counting. When

you type in these lines, replace inverse full stops with black squares (e.g., if nine inverse full stops are shown, type in nine inverse spaces). *Do not* do this with any other lines.

150 Draw rabbit's coat in correct colour on rabbit.

160 PF is the profit so far.

170–180 Redo for other rabbits.

200–250 Write title, wait for rabbit to be selected, refusing illegal entries.

260 Subtract cost of rabbit being kept from profit – you can't sell your rabbit and keep it!

270 Store rabbit's genes in L$.

280–290 Clear screen and repeat for female rabbits.

300–310 Increment generation counter and finish if more than five years. This can be changed as desired.

500–570 Breed eight offspring with characteristics chosen at random from each parent, but making sure always one gene from male, one from female parent, not both of a pair from just one parent, as rabbits do not form clones.

580 Do another generation.

700–790 End of game, showing profit, highscore and offering new game. If new game accepted, all variables reset except highscore, HI.

1000 Go and find rabbit's visible characteristics.

1010–1040 A$, B$, C$, D$ are rabbit's coat colours and these are swapped around so that if a rabbit is patchy, it won't necessarily have patches in the same places as another patchy rabbit.

1050–1060 Put 'ear genes' in X$(1) and X$(2).

1100–1150 Select ear-size according to 'ear genes' and some luck.

1500–2550 Select the actual colours of the rabbit, A$ and D$ according to colour, genes and random chance (i.e., if colour genes are 'dotted' and 'arrowed' the rabbit will be dotted, or patched in dots and arrows. You cannot tell which in advance, but it cannot be grey for instance.

Note that if A$ and D$ are the same, the rabbit has one 1 colour (or is striped) but if A$ and D$ are different, the rabbit is randomly patched. For grey patches, you may use

graphic shift A or graphic shift H, but be consistent. Do not change from one to the other: if you use graphic shift A for one, you *must* use it for all the others or the program won't work properly.

8000–8220 Starting variables.

HI = highscore, L$ contains genes of selected breeding pair of rabbits, K$ contains the possible 'colour genes'.

N$ contains possible 'ear genes'.

GN = generation number.

X$ = ear genes of one offspring.

M$ contains all the genes of all the rabbits in a generation. These are selected for the first generation randomly from K$+N$ in lines 8100–8170.

V = value of rabbits in one generation.

PF = profit, initially 0.

The program returns to line 20.

9000–9110 Rules, omit or expand as desired.

Listing

```
   1 REM RABBITS
  10 GOSUB 9000
  20 FOR Z=1 TO 2
  30 LET CT=0
  40 FOR X=0 TO 15 STEP 15
  50 FOR Y=0 TO 10 STEP 10
  60 LET CT=CT+1
  70 GOSUB 1000
  80 PRINT AT Y,X;"£";V(Z,CT)

 100 PRINT AT Y,X+7;J$;"▐";AT Y+
1,X+7;J$;"█";AT Y+2,X+1;"
▆ ";AT Y+3,X;"▆▆▆▆▆▆▆▆"

 110 PRINT AT Y+4,X;"▆▆▆▆▆▆▆▆
";AT Y+5,X;"▆▆▆▆▆▆▆";AT Y+6,X;
"▆▆ ▆▆"

 120 PRINT AT Y+7,X;"▆  ▀
AT Y+8,X+1;"▆▆   ";CT
 150 PRINT AT Y+2,X+3;B$;AT Y+3,
X+1;C$;AT Y+4,X;C$;A$;B$;AT Y+5,
X;H$;C$;AT Y+6,X+5;B$;A$

 160 LET PF=PF+V(Z,CT)
 170 NEXT Y
 180 NEXT X
```

```
 200 IF Z=1 THEN PRINT AT 20,0;"
MALES";
 210 IF Z=2 THEN PRINT AT 20,0;"
FEMALES";
 220 PRINT ", GENERATION ";GN
 230 PRINT "CHOOSE THE LUCKY RAB
BIT"
 240 LET P$=INKEY$
 250 IF P$>"4" OR P$<"1" THEN GO
TO 240
 260 LET PF=PF-V(Z,VAL P$)
 270 LET L$(Z)=M$(Z,VAL P$)
 280 CLS
 290 NEXT Z

 300 LET GN=GN+1
 310 IF GN>5 THEN GOTO 700

 500 FOR F=1 TO 2
 510 FOR G=1 TO 4
 520 FOR H=1 TO 4
 530 LET M$(F,G,H)=L$(1,H)
 540 IF RND<.5 THEN LET M$(F,G,H
)=L$(2,H)
 550 NEXT H
 560 NEXT G
 570 NEXT F
 580 GOTO 20

 700 CLS
 710 PRINT "AFTER 5 YEARS..."
 720 PRINT
 730 PRINT "YOU HAVE MADE £";PF;
" PROFIT."
 740 IF PF>HI THEN LET HI=PF
 750 PRINT "BEST SO FAR = ";HI
 760 PRINT AT 20,0;"ANOTHER GAME
? (Y/N) "
 770 IF INKEY$="N" THEN STOP
 780 IF INKEY$="" THEN GOTO 760
 790 GOTO 6010

1000 GOSUB 2000
1010 LET B$=A$+D$
1020 IF RND<.5 THEN LET B$=D$+A$
1030 LET C$=B$+B$+B$+B$
1040 IF RND<.5 THEN LET C$=A$+B$
+D$+B$+D$+A$
1050 LET X$(1)=M$(Z,CT,3)
1060 LET X$(2)=M$(Z,CT,4)

1100 LET J$="█"
1110 IF X$(1)=" " OR X$(2)=" " T
HEN LET J$=" "
```

```
1120 IF X$(1)="█" AND X$(2)="█"
THEN LET J$="█"
1130 IF B$="█" AND J$="█" AND R
ND<.5 THEN LET J$="▓"
1140 IF B$="█" AND J$="█" AND R
ND<.4 THEN LET J$="█"
1150 IF RND<.05 THEN LET J$="█"

1500 IF A$<>D$ THEN GOTO 1600
1510 LET WT=10
1520 IF A$="█" OR A$="█" THEN LE
T WT=15
1530 IF A$="█" OR A$="█" THEN LE
T WT=30
1540 GOTO 1610

1600 LET WT=1
1610 IF J$="█" THEN LET WT=WT+2
1620 IF J$="█" THEN LET WT=WT+5
1630 IF J$="█" THEN LET WT=WT+15
1640 IF J$="▓" THEN LET WT=WT+5
1650 IF J$="█" THEN LET WT=WT+8
1660 LET V(Z,CT)=WT

2000 LET E$=M$(Z,CT,1)
2010 LET F$=M$(Z,CT,2)
2020 LET A$=E$
2030 IF E$=F$ AND E$="█" AND RND
<.5 THEN LET A$="█"
2040 LET D$=A$
2050 IF E$=F$ THEN RETURN

2100 IF E$="█" OR F$="█" THEN GO
TO 2300
2110 IF E$="▓" OR F$="▓" THEN GO
TO 2500
2120 IF E$="█" OR F$="█" THEN GO
TO 2200
2150 LET A$="█"
2160 LET D$=A$
2170 IF RND<.5 THEN LET D$="█"
2180 RETURN

2200 LET A$="█"
2210 IF E$="█" OR F$="█" THEN GO
TO 2250
2220 LET D$=A$
2230 IF RND<.5 THEN LET D$="█"
2240 RETURN
2250 LET A$="█"
2260 LET D$=A$
2270 RETURN
```

```
2300 LET A$="█"
2310 LET D$=A$
2320 IF E$="▓" OR F$="▓" THEN GO
TO 2400
2330 IF E$="◀" OR F$="◀" OR E$="
█" OR F$="█" THEN RETURN
2340 IF RND<.5 THEN LET D$="█"
2350 RETURN
2400 LET CH=RND
2410 IF CH<.5 THEN RETURN
2420 LET A$="▓"
2430 IF CH<.5 THEN LET A$="▓"
2440 IF CH>.9 THEN LET A$="▓"
2450 GOTO 2260

2500 LET A$="▓"
2510 LET D$=A$
2520 IF E$="█" OR F$="█" THEN RE
TURN
2530 IF (E$="█" OR F$="█") AND R
ND<.5 THEN LET D$="█"
2540 IF (E$="█" OR F$="█") AND R
ND<.5 THEN LET D$="█"
2550 RETURN

8000 LET H1=0
8010 DIM L$(2,4)
8020 LET K$="█ █ = █"
8030 LET N$="█ █ █"
8040 LET GN=1
8050 DIM X$(2)
8060 DIM M$(2,4,4)
8070 RAND

8100 FOR F=1 TO 2
8110 FOR G=1 TO 4
8120 FOR H=1 TO 2
8130 LET M$(F,G,H)=K$(INT (RND*5
)+1)
8140 LET M$(F,G,H+2)=N$(INT (RND
*3)+1)
8150 NEXT H
8160 NEXT G
8170 NEXT F

8200 DIM V(2,4)
8210 LET PF=0
8220 RETURN

9000 PRINT "RABBITS"
9010 PRINT
```

```
9020 PRINT " YOU ARE A RABBIT BR
EEDER WITH  8 YOUNG RABBITS. EAC
H YEAR, YOU MAY KEEP 2 YOUNG FOR
 BREEDING,"
9030 PRINT "AND SELL THE REST TO
 A PET SHOP. THE PRICE OF A RABB
IT DEPENDS  ON ITS COLOUR AND EA
R SIZE."

9040 PRINT "RABBITS WITH EARS TH
AT ARE THIN OR VERY WIDE COST MO
RE THAN"
9050 PRINT "NORMAL-EARED RABBITS
. THOSE WITH UNUSUAL COLOURING A
RE EXPENSIVE WHILE PATCHY ONES A
RE CHEAP."
9060 PRINT " EACH YEAR, YOU CHOO
SE 1 MALE   AND 1 FEMALE RABBIT
TO BREED THE NEXT GENERATION."
9070 PRINT

9080 PRINT "(PRESS NEWLINE TO ST
ART)"
9090 IF INKEY$<" " THEN GOTO 909
0
9100 CLS
9110 GOTO 8000
```

Base to Decimal Converter

Program runs in 1K

To use

This program converts numbers in any base (binary or above) to decimal. The program first asks for the number you wish to be converted to decimal. Next you input the base in which your number is expressed. After a short wait (up to ten seconds for a long number) the decimal equivalent is given.

Bases larger than base 10 need special digits for decimal figures – ten, eleven, etc. – as 10 in (for example) base 15 is 15 in base 10, 11 is 16 in base 10 and 12 is 17 in base 10. Generally the extra digits used in these situations are capital letters. So counting in, for example, base 12, you write 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, 10, 11 . . . .

The program can accept these letters. So if you wish to know what FF (hexadecimal) is in base 10, input FF for your number, and 16 for the base.

Note that the program is accurate to eight significant figures.

Notes on listing

10–50 Input number and base and clear screen.

60 PRINT beginning of answer.

70 T will be the answer.

100–120 Starting from the right of the number, the first digit is always units, the second is the number × the base, the third is the number × the base squared and so on. So, in base 10, the first digit to the right is units, the second tens, the third hundreds, etc.

The program calculates from the first digit, knowing the length of your number.

200 Answer in base 10 correct to 8 significant figures.

Listing

```
   1 REM   BASE TO DECIMAL
CONVERTER
   2 REM   COPYRIGHT 1982
             S.ROBERT SPEEL

  10 PRINT "INPUT 1ST NUMBER"
  20 INPUT A$
  30 PRINT "WHAT BASE IS IT?"
  40 INPUT B
  50 CLS
  60 PRINT A$;" IN BASE ";B;" ="
  70 LET T=0

 100 FOR F=1 TO LEN A$
 110 IF A$(F)>"0" THEN LET T=T+(
CODE A$(F)-28)*B**(LEN A$-F)
 120 NEXT F

 200 PRINT T;" IN BASE 10"
```

Decimal to Base Converter

Program runs in 1K

To use
This program converts numbers from base 10 to any other base.
First input your number, then the base you wish to convert to.
The program will then convert the number, showing its
calculations.

```
DECIMAL NUMBER = 43151

BASE          NUMBER        REMAINDER
4             43151         3
4             10787         3
4             2696          0
4             674           2
4             168           0
4             42            2
4             10            2
4             2             2

43151 IN BASE 4 IS 22202033
```

The program can cope with bases larger than 10, and uses
letters for the extra digits.

```
DECIMAL NUMBER = 66318

BASE          NUMBER        REMAINDER
16            66318         E
16            4144          0
16            259           3
16            16            0
16            1             1

66318 IN BASE 16 IS 1030E
```

Notes on listing

100 A$ will be the number.

110–140 Input decimal number and desired base.

210–220 Print decimal number and column headings.

250–270 Calculated digits by dividing number by base and keeping remainder. Print number, base and remainder in columns.

280 Repeat if unfinished.

300–310 Print answer.

Listing

```
   1 REM DEC TO BASE CONVERTER
     COPYRIGHT S.R.SPEEL

 100 LET A$=""
 110 PRINT "NUMBER?"
 120 INPUT A
 130 PRINT "BASE TO CONVERT TO?"
 140 INPUT B

 200 CLS
 210 PRINT "DECIMAL NUMBER = ";A
 220 PRINT "BASE";TAB 10;"NUMBER
";TAB 20;"REMAINDER"
 230 LET C=A

 250 LET A$=CHR$ (28+A-INT (A/B)
*B)+A$
 260 PRINT B;TAB 10;A;TAB 20;A$(
1)
 270 LET A=INT (A/B)
 280 IF A>0 THEN GOTO 250

 300 PRINT
 310 PRINT C;" IN BASE ";B;" IS
";A$
```

Quadratic Equation Solver

Listing occupies 1.4K
Program runs in 2.5K

This program solves any quadratic equation with real roots using the quadratic formula.

It also writes out step by step the working out by which the answer is achieved. This means that a problem can be typed in, and then the answer and working copied directly into a maths homework book . . . or for those with scruples, it becomes a way of really checking the maths homework.

The program asks you to input A, B and C for the general quadratic

$$AX^2 + BX + C = 0$$

and starts writing up the solution immediately.

Since many quadratic equations have no real solution, especially those in maths homework questions, if such an equation is entered the program will say that it is unsolvable. It also gives an explanation why.

```
TO SOLVE THE QUADRATIC:-
  3XX+2X+6=0
 USING THE QUADRATIC FORMULA,

      +
X=-2-   2*2-(4*3*6)
      ────────────────
           6

      +
X=-2-   4-72
      ──────────
          6
```

SINCE 4-72=-68,
THIS EQUATION HAS NO REAL ROOTS,
 AS THE SQUARE ROOT OF -68
IS IMAGINARY.

TO SOLVE THE QUADRATIC:-
  -13XX+28X+16.452=0
 USING THE QUADRATIC FORMULA,

$$X = \frac{-28 \overset{+}{-} \sqrt{28*28-(4*-13*16.452)}}{-26}$$

$$X = \frac{-28 \overset{+}{-} \sqrt{784--855.504}}{-26}$$

$$X = \frac{-28+40.49}{-26} \text{ OR } \frac{-28-40.49}{-26}$$

HENCE X=-0.481 OR 2.634

TO SOLVE THE QUADRATIC:-
  2XX+-1X+-3=0
 USING THE QUADRATIC FORMULA,

$$X = \frac{--1 \overset{+}{-} \sqrt{-1*-1-(4*2*-3)}}{4}$$

$$X = \frac{--1 \overset{+}{-} \sqrt{1--24}}{4}$$

$$X = \frac{1+5}{4} \text{ OR } \frac{1-5}{4}$$

HENCE X=1.5 OR -1

Notes on listing

100–170 'Introduction' prompts for A,B,C in equation.

The computer uses AXX instead of AX∗∗2 since it looks less confusing. This is explained in the program.

A = X squared coefficient

B = X coefficient

C = constant

180 Clear screen before starting to solve equation.

190 A$ contains the line we will use to underline things, for square root signs and long division signs.

200–520 Work out answers, writing down each step. All the LEN(STR$A+LEN STR$B . . . information looks very complicated. Most of it is simply to ensure that the square root signs and the long division signs are the right length. This length varies according to how well the equation works out; one with whole number solutions will need only short lines, while an equation with awkward roots such as 3.263 and 2.964 will need longer lines.

Compare the examples given and see how much the lengths vary.

Lines such as 450 round down the number to three decimal places.

If you wish to round *off* rather than round *down* change all the relevant lines to INT(0.5+ . . .), so that line 450 would become:

$$450 \text{ LET XA} = \text{INT}(0.5 + 1000 * (-B + D)/(2 * A))/1000$$

Note that in lines 350 to 370, we seem to waste a line, since 350 and 370 could be combined into one line. However, the ZX81 would crash if it tried to solve the square root of a negative number. Thus 360 checks for this and sends control to line 600 if a negative number is to be rooted. ·

Note that if there is just one solution, the computer will not print it twice, but just once. One solution is obtained when the number inside the square root sign is zero. Line 480 checks for this and alters B$ (the answer) accordingly.

600–630 If no real roots, say so and explain why.

Listing

```
   1 REM QUADRATIC EQUATION
SOLVER
   2 REM     COPYRIGHT 1982
   3 REM     S.ROBERT SPEEL
 100 PRINT " QUADRATIC EQUATION
SOLVER"
 110 PRINT
 120 PRINT "FOR THE QUADRATIC AX
X+BX+C=0"
 130 PRINT " (WHERE XX IS X SQUA
RED)"
 140 PRINT "INPUT A,B,C, WITH NE
WLINE AFTER EACH."
 150 INPUT A
 160 INPUT B
 170 INPUT C

 180 CLS
 190 LET A$="                    "

 200 PRINT "TO SOLVE THE QUADRAT
IC:-"
 210 PRINT " ";STR$ A;"XX+";STR
$ B;"X+";STR$ C;"=0"
 220 PRINT " USING THE QUADRATIC
FORMULA,"
 230 PRINT

 240 LET D=LEN (STR$ A+STR$ B+ST
R$ B+STR$ C)+7
 250 PRINT TAB 3+LEN STR$ B;"+ "
;A$( TO D)
 260 PRINT "X=-";B;" ";B;"*";B;
"-(4*";A;"*";C;")"
 270 PRINT TAB 2;A$( TO D+5)
 280 PRINT TAB LEN A$( TO D+4) /2
;2*A
 290 PRINT

 300 PRINT TAB 3+LEN STR$ B;"+ "
;A$( TO LEN (STR$ (B*B)+STR$ (4*
A*C))+1)
 310 PRINT "X=-";B;" ";B*B;"-";
4*A*C
 320 PRINT TAB 2;A$( TO 4+LEN (S
TR$ B+STR$ (B*B)+STR$ (4*A*C)))
 330 PRINT TAB 5+LEN STR$ B;2*A
 340 PRINT
 350 LET D=B*B-4*A*C
 360 IF D<0 THEN GOTO 600
 370 LET D=INT (1000*SQR D) /1000
```

```
 400 PRINT "X=";-B;"+";D;" OR ";
-B;"-";D
 410 LET E=LEN (STR$ -B+STR$ D)+
1
 420 PRINT TAB 2;A$( TO E);"
";A$( TO E)
 430 PRINT TAB (2+E)/2;A*2;TAB 4
+E+(2+E)/2;2*A
 440 PRINT
 450 LET XA=INT (1000*(-B+D)/(2*
A))/1000
 460 LET XB=INT (1000*(-B-D)/(2*
A))/1000
 470 LET B$="X="+STR$ XA+" OR "+
STR$ XB
 480 IF D=0 THEN LET B$="X="+STR
$ XA

 500 PRINT " HENCE ";B$
 510 PRINT A$( TO LEN B$+7)
 520 STOP

 600 PRINT
 610 PRINT "SINCE ";B*B;"-";4*A*
C;"=";D;" ,"
 620 PRINT "THIS EQUATION HAS NO
 REAL ROOTS, AS THE SQUARE ROOT
OF ";D
 630 PRINT "IS IMAGINARY."
```
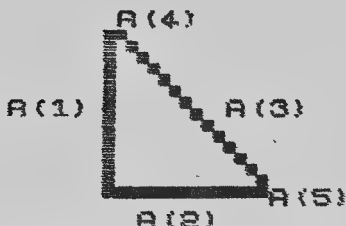
Notes on conversion for Spectrum

The program can be typed in directly from the ZX81 listing.
However, improvements can be made. The thick dividing line
and square root sign can be replaced with thinner lines using
DRAW. The program can be reduced by using multi-line state-
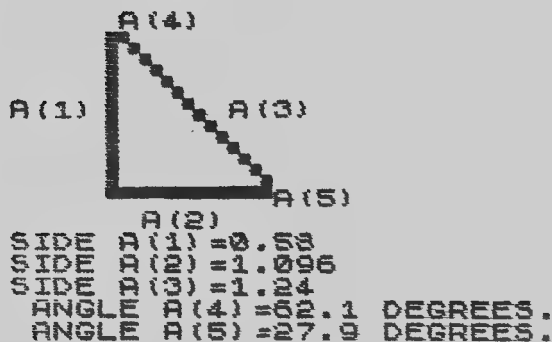ments.

Right-angled Triangle Solver

```
A(4)        RIGHT ANGLED
             TRIANGLE SOLVER.
A(1)           A(3)

               A(5)
     A(2)

INPUT SIDE A(1)...0
INPUT SIDE A(2)...0
INPUT SIDE A(3)...1.24
INPUT ANGLE A(4)...0
INPUT ANGLE A(5)...27.9


A(4)

A(1)           A(3)

               A(5)
     A(2)
 USING "SUM OF ANGLES IN A
TRIANGLE=180 DEGREES",
A(4)=90-A(5)
A(4)=62.1
USING SIN A(4)=A(2)/A(3)
A(2)=A(3)*SIN A(4)
A(2)=1.0958694
 USING PYTHAGORAS,
A(1)**2=A(3)**2-A(2)**2
A(1)=SQR (1.5376-1.2009297)
A(1)=0.58023297
(PRESS NEWLINE TO SEE ANSWERS)
```

```
          A(4)

A(1)           A(3)

                    A(5)
        A(2)
SIDE  A(1) =0.58
SIDE  A(2) =1.096
SIDE  A(3) =1.24
  ANGLE  A(4) =62.1 DEGREES.
  ANGLE  A(5) =27.9 DEGREES.
```

Listing occupies 2.8K
Program runs in 3.8K

To use

This program will solve any right-angled triangle. It also gives
full working out stages, printing out which formulae are used
for each stage. This program, therefore, can be used to check
working, or as a quicker way of doing homework.

On running, the computer draws a right-angled triangle and
asks for the sides and angles that you know. The usual notation
of angles A,B,C and sides a,b,c has been avoided as lower-case
letters are not possible. Instead, the sides are labelled A(1),
A(2) and A(3) and the angles (not the right angle) A(4) and
A(5).

You enter only two of the values at least one of which must
be a side, since a triangle where only the angles are known is
unsolvable. Do not enter more than two values, even if you
have them available. For the other three values enter 0. The
screen then clears and the triangle is redrawn, with full working
out underneath, together with the answers. On pressing
NEWLINE, the triangle is again drawn, together with all the
correct sides and angles in a short list.

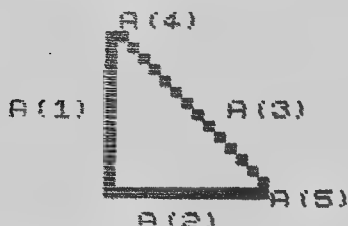Note that final answers are given to four significant figures.

```
      A(4)      RIGHT ANGLED
                 TRIANGLE SOLVER.

 A(1)        A(3)


            A(5)
     A(2)

INPUT SIDE A(1)...84.27
INPUT SIDE A(2)...36
INPUT SIDE A(3)...0
INPUT ANGLE A(4)...0
INPUT ANGLE A(5)...0


      A(4)



 A(1)        A(3)


            A(5)
     A(2)
 USING PYTHAGORAS,
A(3)**2=A(1)**2+A(2)**2
A(3)=SQR (7101.4329+1296)
A(3)=91.637508
USING SIN A(4)=A(2)/A(3),
A(4)=23.132091 DEGREES
 USING "SUM OF ANGLES IN A
TRIANGLE=180 DEGREES",
A(5)=90-A(4),
A(5)=66.867909
(PRESS NEWLINE TO SEE ANSWERS)


      A(4)



 A(1)        A(3)


            A(5)
     A(2)
SIDE A(1)=84.27
SIDE A(2)=36
SIDE A(3)=91.638
 ANGLE A(4)=23.132 DEGREES.
 ANGLE A(5)=66.868 DEGREES.
```
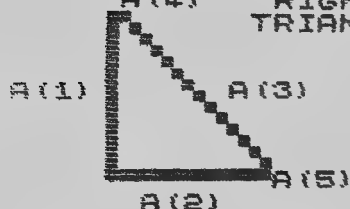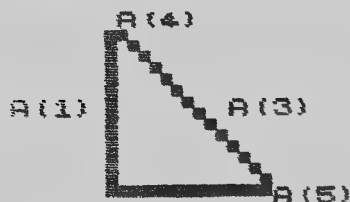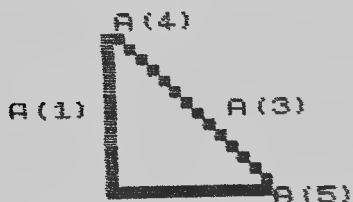
RIGHT ANGLED
TRIANGLE SOLVER.

```
INPUT SIDE A(1)...0
INPUT SIDE A(2)...141
INPUT SIDE A(3)...0
INPUT ANGLE A(4)...66.66
INPUT ANGLE A(5)...0
```



```
 USING "SUM OF ANGLES IN A
TRIANGLE=180 DEGREES",
A(5)=90-A(4),
A(5)=23.34
USING COS A(5)=A(2)/A(3),
A(3)=A(2)/COS A(5)
A(3)=153.56633
 USING PYTHAGORAS,
A(1)**2=A(3)**2-A(2)**2
A(1)=SQR (23582.618-19881)
A(1)=60.840922
(PRESS NEWLINE TO SEE ANSWERS)
```



```
SIDE A(1)=60.841
SIDE A(2)=141
SIDE A(3)=153.566
 ANGLE A(4)=66.66 DEGREES.
 ANGLE A(5)=23.34 DEGREES.
```

Notes on listing

Note that the computer works not in degrees, but in radians. There are two PI radians in a circle, i.e., one radian is about 57.3°. This means that any inputted values must be converted into radians before the computer works them out. Also, whenever an angle is solved or printed, it must first be converted back into degrees.

Note that ASN (as in line 810) is written ARCSIN on the keyboard. Also, in line 600, the double inverted commas character is used (referred to as 'quote image character' in ZX81 manual but often known as 'double inverts'). Do not put two inverted comma characters, it won't work.

10 Goto start of program.

50 Print triangle.

100 If A(4) AND A(5) means If A(4) and A(5) are logically true, which is another way of saying IF A(4) < > 0 AND A(5) < > 0 but quicker. If only the two angles are known, goto 450, as triangle is insoluble.

110 If either angle is known, goto 200.

120–140 No angles are known. Therefore two sides are known. Find third side by Pythagoras' theorem. IF NOT A(1) means IF A(1) = 0.

150 If the two angles are now known, then goto 'Triangle solved' in line 300.

160 Goto find angle A(4) sub-routine.

170 Goto find angle A(5) sub-routine.

180 Goto 'Triangle solved' in line 300.

200–240 One side, one angle known.

200 Gosub find second angle routine, line 500.

220–240 Find second side of triangle and finish triangle in lines 120–150.

300–400 Answers. Clear screen on command, redraw triangle and print three sides and two angles underneath.

Note radian answers for angles are converted to degrees by multiplying by 180∗PI.

450–460 Triangle insoluble because only angles are given.

500–560 Find angle A(5) from angle A(4) or vice versa. Logic

is used to determine which angle must be found. So if A(5) = 0 is logically true, the angle to be found is A(5).

540 The angle to be found is 90° minus the known angle. 90° is .5 PI radians.

600–640 Find side A(1) by Pythagoras' theorem.

650–690 Find side A(2) by Pythagoras' theorem.

700–740 Find side A(3) by Pythagoras' theorem.

800–850 Find angle A(4) from sides A(2) and A(3).

850–890 Find side A(2) from side A(1) and angle A(5).

900–940 Find side A(3) from angle A(5) and side A(2).

950–990 Find side A(2) from side A(3) and angle A(4).

8000–8220 Start of program, gosub draw triangle, request inputs of various sides/angles. Convert angles from degrees to radians.

8500–8570 Sub-routine to draw right-angled triangle and label sides and angles.

Listing

```
   1 REM RIGHT ANGLED TRIANGLE S
OLVER
   2 REM      COPYRIGHT 1982
              S.ROBERT SPEEL
  10 GOSUB 8000
  50 GOSUB 8500
 100 IF A(4) AND A(5) THEN GOTO
450
 110 IF A(4) OR A(5) THEN GOTO 2
00
 120 IF NOT A(1) THEN GOSUB 600
 130 IF NOT A(2) THEN GOSUB 650
 140 IF NOT A(3) THEN GOSUB 700
 150 IF A(4) AND A(5) THEN GOTO
300
 160 GOSUB 800
 170 GOSUB 500
 180 GOTO 300

 200 GOSUB 500
 220 IF A(1) THEN GOTO 850
 230 IF A(2) THEN GOTO 900
 240 GOTO 950

 300 PRINT "(PRESS NEWLINE TO SE
E ANSWERS)"
 310 IF INKEY$<="" THEN GOTO 310
 320 CLS
```

```
330 GOSUB 8500
340 FOR F=1 TO 3
350 PRINT "SIDE A(";F;")=";INT
(1000*A(F)+.5)/1000
360 NEXT F
370 FOR F=4 TO 5
380 PRINT " ANGLE A(";F;")=";IN
T (1000*A(F)*180/PI+.5)/1000;" D
EGREES."
390 NEXT F
400 STOP

450 PRINT "THIS TRIANGLE CANNOT
 BE SOLVED."
460 STOP
500 PRINT " USING ""SUM OF ANGL
ES IN A          TRIANGLE=180 DEGREE
S"","
510 LET RS=(A(5)=0)*4+(A(5)>0)*
5
520 LET SR=(A(4)=0)*4+(A(4)>0)*
5
530 PRINT "A(";SR;")=90-A(";RS;
")"
540 LET A(SR)=.5*PI-A(RS)
550 PRINT "A(";SR;")=";A(SR)*16
0/PI
560 RETURN

600 PRINT " USING PYTHAGORAS,
          A(1)**2=A(3)**2-A(2)
**2"
610 PRINT "A(1)=SQR (";A(3)**2;
"-";A(2)**2;")"
620 LET A(1)=SQR (A(3)**2-A(2)*
*2)
630 PRINT "A(1)=";A(1)
640 RETURN

650 PRINT " USING PYTHAGORAS,
          A(2)**2=A(3)**2-A(1)
**2"
660 PRINT "A(2)=SQR (";A(3)**2;
"-";A(1)**2;")"
670 LET A(2)=SQR (A(3)**2-A(1)*
*2)
680 PRINT "A(2)=";A(2)
690 RETURN
700 PRINT " USING PYTHAGORAS,
          A(3)**2=A(1)**2+A(2)
**2"
710 PRINT "A(3)=SQR (";A(1)**2;
"+";A(2)**2;")"
```

```
 720 LET A(3)=SQR (A(1)**2+A(2)*
*2)
 730 PRINT "A(3)=";A(3)
 740 RETURN

 800 PRINT "USING SIN A(4)=A(2)/
A(3),"
 810 LET A(4)=ASN (A(2)/A(3))
 820 PRINT "A(4)=";A(4)*180/PI;"
DEGREES"
 830 RETURN

 850 PRINT "USING TAN A(5)=A(1)/
A(2),"
 860 PRINT "A(2)=A(1)/TAN A(5)"
 870 LET A(2)=A(1)/TAN A(5)
 880 PRINT "A(2)=";A(2)
 890 GOTO 120

 900 PRINT "USING COS A(5)=A(2)/
A(3),"
 910 PRINT "A(3)=A(2)/COS A(5)"
 920 LET A(3)=A(2)/COS A(5)
 930 PRINT "A(3)=";A(3)
 940 GOTO 120

 950 PRINT "USING SIN A(4)=A(2)/
A(3)"
 960 PRINT "A(2)=A(3)*SIN A(4)"
 970 LET A(2)=A(3)*SIN A(4)
 980 PRINT "A(2)=";A(2)
 990 GOTO 120

6000 DIM A(5)
8010 PRINT TAB 12;"RIGHT ANGLED"
;TAB 10;" TRIANGLE SOLVER."
8020 PRINT
8030 GOSUB 8500
8040 PRINT

8100 FOR F=1 TO 3
8110 PRINT "INPUT SIDE A(";F;")"
;
8120 INPUT A(F)
8130 PRINT "...";A(F)
8140 NEXT F
8150 FOR F=4 TO 5
8160 PRINT "INPUT ANGLE A(";F;")"
";
8170 INPUT A(F)
8180 PRINT "...";A(F)
8190 LET A(F)=A(F)*PI/180
8200 NEXT F
8210 CLS
8220 RETURN
```

```
8500 PRINT AT 0,5;"A(4)"
8510 FOR F=1 TO 7
8520 PRINT AT F,4;"█";AT F,F+4;"
█"
8530 NEXT F
8540 PRINT TAB 4;"███████A(5)"
8550 PRINT AT 4,0;"A(1)";TAB 10;
"A(3)"
8560 PRINT AT 9,6;"A(2)"
8570 RETURN
```

Compound Interest

Listing occupies 2.3K
Program runs in 3.3K

To use
This program calculates compound interest on any sum of money invested at any interest rate for any number of years. On running, you are asked to input first the interest rate then the times when the interest is calculated, then the initial investment.

The program also allows you to take out a fixed sum, or invest a fixed sum each month/year or specified number of months.

The program asks for how much you want to deposit regularly and at what monthly intervals. If you do not want to invest any regular further amounts, input zero for both these values. You are then given a choice between seeing the monetary situation after a specified number of years, or seeing how the cash builds up over a period of ten years, year by year.

For example, you wish to invest £1000 in one of two banks: Bank 1 offers a compound interest rate of 12.5 per cent with interest added on every six months, and Bank 2 offers a compound interest of 12.8 per cent added on annually. Which bank should you choose to invest in to obtain the highest return after nine years?

Run the program and input 12.5, 6, 1000, 0, 0 (0, 0 here because no additional money invested or withdrawn), in answer to the questions. Then input A. The screen will show the following.

```
GROWTH OF CAPITAL

INITIAL DEPOSIT =£1000
AT 12.5 PERCENT CALCULATED EACH
   6 MONTHS.

YEAR DEPOSIT INTEREST NEW DEPOS.
1    1000     128.9    0
2    1128.9   145.5    0
3    1274.4   164.3    0
4    1438.7   185.5    0
5    1624.2   209.4    0
6    1833.5   236.4    0
7    2069.9   266.8    0
8    2336.7   301.2    0
9    2637.9   340      0
10   2978     383.9    0

FURTHER 10 YEARS?(PRESS Y/N)
```

Stop the program and repeat for Bank 2, inputs being 12.8, 12, 1000, 0, and 0. Input A to get the following output.

```
GROWTH OF CAPITAL

INITIAL DEPOSIT =£1000
AT 12.8 PERCENT CALCULATED EACH
   12 MONTHS.

YEAR DEPOSIT INTEREST NEW DEPOS.
1    1000     128      0
2    1128     144.4    0
3    1272.4   162.9    0
4    1435.2   183.7    0
5    1619     207.2    0
6    1826.2   233.8    0
7    2059.9   263.7    0
8    2323.6   297.4    0
9    2621     335.5    0
10   2956.5   378.4    0

FURTHER 10 YEARS?(PRESS Y/N)
```

This shows that at the beginning of, for example, year five (= end of year four) you would have £1624.20 in Bank 1, or £1619.00 in Bank 2. At the beginning of the tenth year, you would have £2978.00 in Bank 1, or £2956.50 in Bank 2. Clearly, if interest rates remain constant, Bank 1 is the better choice.

Another example

You are considering investing in a building society. For an initial investment of £1500, and a further £100 invested every six months, they offer a guaranteed 10.2 per cent compound interest, calculated quarterly. You want to know how much you would have after eight years, and of this amount, how much would be profit; that is, the total interest earned so far.

Run the program, inputting 10.2, 3 (3 months = quarterly), 1500, 100, 6 (£100 every 6 months) and then input B. Then press 8 to prompt 'how many years . . . ?' This gives the following output, showing total in account after eight years and profit if you withdraw at the end of the eighth year.

```
 GROWTH OF CAPITAL

INITIAL DEPOSIT =£1500
AT 10.2 PERCENT CALCULATED EACH
 3 MONTHS.
DEPOSIT OF £100 EVERY 6 MONTHS.

AFTER 8 YEARS...

TOTAL MONEY INVESTED =£5755.33

TOTAL PROFIT =3855.3284
 (PRESS NEWLINE TO RETURN)
```

If you want to withdraw money regularly, simply input a negative value to the prompt "INPUT REGULAR DEPOSIT . . ." Note that if the withdrawal amount is larger than the interest, the capital invested decreases and will eventually become a debit . . . . This would not be possible in reality.

Notes on listing
100–290 Input various variables to prompts.
   A = interest rate
   B = when interest computed (in months)
   C = initial investment
   D = subsequent investments/withdrawals
   E = interval between subsequent investments/withdrawals

300–450 Choices of whether to see interest calculated year by year, or total after a number of years, or stop program or do new problem.

900–1240 Main loop for calculating compound interest. This is done in FAST mode.

920–980 Print title and details of investment.

1000–1020 Y = number of years being considered

DP = total investment at beginning of each year

TM = total months considered. This is useful when extra money is added every fifteen months, or interest calculated bi-annually, etc.

1030 EC = total of years being considered. Either this is set by user (B mode) or set at 10 (A mode).

1100 Print column titles if in A mode.

1110 For F = year 1 to last year (1 to 10 in A mode, or subsequently 11–20, 21–30 etc.).

1120–30 ND = newly deposited money this year

RS = interest accumulated this year

1140 Interest is calculated monthly.

1150 Increment 'total month counter'.

1160 If interest should be added this month, calculate compound interest and add it.

1170 If money put in or taken out this month, do it. Note that TM/B = INT (TM/B) only when interest is due, and similarly for E. This allows odd values, such as money removed every seven months, still to work.

1180 Do again for all twelve 12 months.

1200 Print list of yearly values if in A Mode.

1210 Increase DP by interest earned and deposits made that year. Note D is unchanged, so other calculations can be made later if necessary (i.e., in B mode).

1220 Repeat for 10 years (A mode) or inputted value (B mode).

1230 Go back to SLOW mode so that answers can be shown.

1240 If in B mode, goto 1500 and print answers.

1300–1380 Show table of interest, etc., and offer another ten years' viewing. If accepted, return to fast mode and repeat, without resetting, TM (total months), DP (money invested

so far) and Y (years considered). Otherwise goto choice loop
at 300.

1400–1420 Select how many years you want done in B mode.
Do loop in 900.

1500–1570 Results of B mode, total money invested and total
profit after EC years. Then return to choice loop.

Listing

```
  10 REM COMPOUND INTEREST
  20 REM      COPYRIGHT 1982
             S.ROBERT SPEEL
 100 CLS
 110 PRINT "COMPOUND INTEREST"
 120 PRINT
 130 PRINT "INPUT THE INTEREST R
ATE PERCENT, IE 12.5 PERCENT IS
ENTERED AS  12.5"
 140 INPUT A
 150 PRINT TAB 10;A
 160 LET A=A/100

 170 PRINT "INPUT INTERVAL BETWE
EN INTEREST CALCULATIONS IN MONT
HS, SO IF  THE INTEREST IS 1/2 Y
EARLY, INPUT 6."
 180 INPUT B
 190 PRINT TAB 10;B
 200 PRINT "INPUT INITIAL DEPOSI
T"
 210 INPUT C
 220 PRINT TAB 10;"£";C

 230 PRINT "INPUT REGULAR DEPOSI
T (IF NONE  THEN INPUT 0)"
 240 INPUT D
 250 PRINT TAB 10;"£";D
 260 PRINT "INPUT INTERVAL BETWE
EN DEPOSITS (IF NONE INPUT 0)"
 270 INPUT E
 280 PRINT TAB 10;E
 290 IF E=0 THEN LET E=13

 300 CLS
 310 PRINT " YOU MAY:-"
 320 PRINT
 330 PRINT "A) SEE GROWTH OF CAP
ITAL OVER A NUMBER OF YEARS,"
 340 PRINT "B) SEE SITUATION AFT
ER A CERTAIN NUMBER OF YEARS."
```

```
  350 PRINT "C) DO NEW CALCULATIO
N."
  360 PRINT "D) STOP PROGRAM."

  400 LET A$=INKEY$
  410 IF A$="A" THEN GOTO 900
  420 IF A$="B" THEN GOTO 1400
  430 IF A$="C" THEN RUN
  440 IF A$="D" THEN STOP
  450 GOTO 400
  900 FAST
  910 CLS
  920 PRINT " GROWTH OF CAPITAL"
  930 PRINT "▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
  940 PRINT
  950 PRINT "INITIAL DEPOSIT =£";
C
  960 PRINT "AT ";A*100;" PERCENT
CALCULATED EACH    ";B;" MONTHS
"
  970 IF D>0 THEN PRINT "DEPOSIT
OF £";D;" EVERY ";E;" MONTHS."
  980 PRINT

 1000 LET Y=0
 1010 LET DP=C
 1020 LET TM=0
 1030 IF A$="A" THEN LET EC=Y+10
 1100 IF A$="A" THEN PRINT "YEAR
DEPOSIT INTEREST NEW DEPOS."
 1110 FOR F=Y+1 TO EC
 1120 LET ND=0
 1130 LET RS=0

 1140 FOR G=1 TO 12
 1150 LET TM=TM+1
 1160 IF TM/B=INT (TM/B) THEN LET
RS=RS+(DP+ND+RS)*A*B/12
 1170 IF TM/E=INT (TM/E) THEN LET
ND=ND+D
 1190 NEXT G

 1200 IF A$="A" THEN PRINT F;TAB
6;INT (DP*10+.5)/10;TAB 14; INT (
RS*10+.5)/10;TAB 25;ND
 1210 LET DP=DP+ND+RS
 1220 NEXT F
 1230 SLOW
 1240 IF A$="B" THEN GOTO 1500

 1300 PRINT
 1310 PRINT "FURTHER 10 YEARS?(PR
ESS Y/N)"
```

```
1320 IF INKEY$="N" THEN GOTO 300
1330 IF INKEY$="Y" THEN GOTO 135
0
1340 GOTO 1320
1350 LET Y=Y+10
1360 CLS
1370 FAST
1380 GOTO 1030

1400 PRINT "AFTER HOW MANY YEARS
 WOULD YOU  LIKE TO SEE INVESTME
NT?"
1410 INPUT EC
1420 GOTO 900

1500 PRINT "AFTER ";EC;" YEARS..
."
1510 PRINT
1520 PRINT "TOTAL MONEY INVESTED
=£";INT (DP*100+.5)/100
1530 PRINT
1540 PRINT "TOTAL PROFIT =";DP-C
-D*E/12*EC
1550 PRINT " (PRESS NEWLINE TO R
ETURN)"
1560 IF INKEY$<="" THEN GOTO 156
0
1570 GOTO 300
```

Bar Graph

Listing occupies 1.0K
Program runs in 2.1K

To use
The program is very simple and self-explanatory. Maximum
and minimum data are the upper and lower limits of the Y axis
on the graph that you want to have shown.

As in all bar graphs, the idea is to show overall trends, rather
than exact values. This is useful for graphs showing
temperature, domestic electricity consumption, club accounts
and other similar data.

The graph takes monthly values, but these can be changed
to annual, weekly or other intervals. So, if, say, annual values
over 15 years were desired for a club membership record, the
following changes would be made.

```
  20 LET B$="19601961196219631963
4196519661967196819691970197119
1973119731974"
  30 DIM B(15)
 160 PRINT "NOW INPUT YEARLY DAT
A"
 200 FOR F=0 TO 14
 420 PRINT AT 20,8;"1960.5...70.
..5"
 500 FOR F=1 TO 15
```

Notes on listing
10–20 DIM string containing names of data.
30 B will hold data.
100–150 Input title of graph, and maximum and minimum
values shown.
160–260 Input data, and if data too large or small, then re-
input it.

250 Convert data to number equivalent to a certain 'height' on screen, so maximum value is 15 squares high, minimum is 0.

300–330 Print title and underline with line right length.
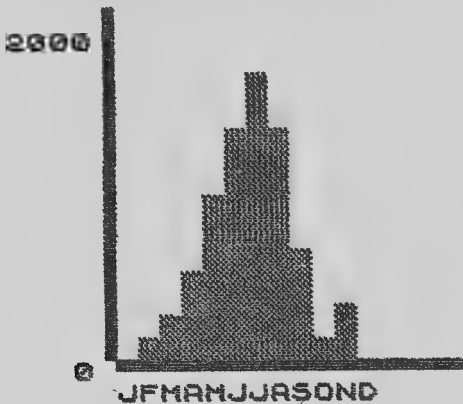
340–370 Print X and Y axes.

400–420 Print Y maximum and minimum values, and initials of months.

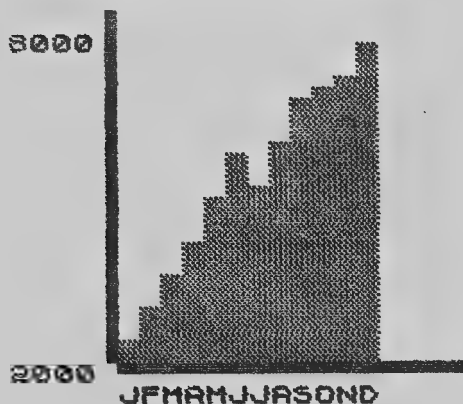**500–570** Print each bar separately. Note use of ½ grey square to give higher accuracy.

```
INPUT GRAPH TITLE
INPUT MAX DATA
INPUT MIN DATA
NOW INPUT MONTHLY DATA
JAN. 10
FEB. 134
MAR. 267
APR. 563
MAY. 1000
JUN. 1407
JUL. 1769
AUG. 1400
SEP. 670
OCT. 200
NOV. 345
DEC. 10
```


MONSOON RAINFALL

BAR GRAPH

WIDGETS SOLD 1962

Listing

```
   1 REM BAR GRAPH
  10 DIM B$(48)
  20 LET B$="JAN.FEB.MAR.APR.MAY
.JUN.JUL.AUG.SEP.OCT.NOV.DEC."
  30 DIM B(12)
 100 PRINT "INPUT GRAPH TITLE"
 110 INPUT A$
 120 PRINT "INPUT MAX DATA"
 130 INPUT MAX
 140 PRINT "INPUT MIN DATA"
 150 INPUT MIN
 160 PRINT "NOW INPUT MONTHLY DA
TA"
 200 FOR F=0 TO 11
 210 PRINT B$(F*4+1 TO F*4+4);"
";
 220 INPUT DAT
 230 IF DAT<MIN OR DAT>MAX THEN
GOTO 220
 240 PRINT DAT
 250 LET B(F+1)=(DAT-MIN)/(MAX-M
IN)*15
 260 NEXT F
 270 CLS
```

```
300 PRINT AT 0,3;A$
310 FOR F=1 TO LEN A$
320 PRINT AT 1,F+2;"■";
330 NEXT F
340 FOR F=0 TO 15
350 PRINT AT 18-F,5;"▐"
360 PRINT AT 19,F+6;"◢"
370 NEXT F
400 PRINT AT 4,5-LEN STR$ MAX;M
AX
410 PRINT AT 19,5-LEN STR$ MIN;
MIN
420 PRINT AT 20,6;"JFMAMJJASOND
"
500 FOR F=1 TO 12
510 FOR G=18 TO 3 STEP -1
520 IF B(F)>=1 THEN PRINT AT G,
F+5;"▓"
530 IF B(F)<1 AND B(F)>=.5 THEN
PRINT AT G,F+5;"▒"
540 IF B(F)<=0 THEN GOTO 570
550 LET B(F)=B(F)-1
560 NEXT G
570 NEXT F
```

Conversion

Conversions and the Printer

When you want to convert, for instance, inches into centimetres, a simple program such as:

```
10 INPUT INS
20 PRINT INS, INS * 2.54
```

would do this. However, it is much simpler to do this with a pocket calculator, or even with pen and paper. So what can the ZX81 do with mathematical functions which a calculator can't? The answer is – lists. A list of conversions of inches to centimetres, and vice versa can be very useful. This is where the ZX Printer comes in.

Using a FOR-NEXT loop, a table of conversions can be printed out and then a print out made using COPY. When you consider it, all conversion tables are very much the same. They all have a column of figures, say for degrees centigrade, with a second column, adjacent, giving equivalent values for degrees fahrenheit. The values for the first column are usually in a definite order, going up in regular steps such as 1, 2, 3, 4 . . . or 20, 40, 60, 80, etc. The degree of accuracy in the conversion will vary enormously, but the conversions to eight decimal places and those to the nearest hundred, still involve the same two columns of figures. This next program is for use with the printer. It will print out columns for most conversions in any steps, accurate to up to eight decimal places. All you need to know is the conversion equation and just what range of values you want to know.

To use

This is best shown by taking an example.

A student wants temperature conversion tables for 'O' level science subjects. He needs a very good conversion from °C to °F for typical room temperatures (about 15–25°C), a reasonably accurate – to nearest degree – conversion from 0–15°C and 25–100°C, and a reasonable idea of °F for 100–500°C. He would also like a rough idea of equivalent temperatures from 500–2000°C.

If we divide these requirements into a list:

Temperature	Step between values shown	Accuracy of °F
0–14°C	each 1°C	1 decimal place
15–25°C	each .5°C	3 decimal places
26–100°C	each 1°C	1 decimal place
110–500°C	each 10°C	no decimal places
550–2000°C	each 50°C	no decimal places

When the program is run, it asks for the title, in this example it would be CENTIGRADE TO FAHRENHEIT.

Next it asks for the heading to Initial Value – input CENT., and the heading for converted value – FAHR. Next you input the conversion factor as an equation.

To get from centigrade to fahrenheit, the centigrade value is multiplied by 9/5, and 32 is then added. You have to put in the equation with F as the unconverted value. In this case, you would input $F * 9/5 + 32$. (Another example, for millimetres to inches, the conversion factor is mm $\times 25.4$, so you would input $F * 25.4$.)

When the program asks for how many scales, in the above example (CENTIGRADE TO FAHRENHEIT) you would put 5. The computer then asks you about each scale, asking for starting and ending values, step between values, and decimal places required.

For the 0–14°C scale therefore: starting value = 0, end value = 14, step = 1 and decimal places = 1; and for 15–25°C, starting value = 15, end value = 25, step = .5 and decimal places = 3.

This would be continued for the remaining three scales and then the list would be printed on the screen, with an error message at the bottom to indicate that there is no more room on the screen.

Press COPY to get a print out, and then CONT. Another screenful of conversions will appear. When this has been COPYed you will see that it joins up smoothly with the first page.

Repeat this until the whole list has been printed out on paper.

The reason for the stopping of the program after each page (if more than one) and the manual entry of COPY and CONT, is so that you can examine each page before printing it out on paper. If you find that you actually need different steps or you need to make a change, you can start again by pressing RUN.

However, if you do not want all this stopping and starting, then just change all the PRINT statements in lines 20–240 (*not* those after this point) to LPRINT statements. Do not rewrite each line, just EDIT each line, move the cursor past the PRINT, RUBOUT the PRINT and put LPRINT instead.

```
    1 REM CONVERSION
   10 GOSUB 8000
   20 LPRINT A$
   30 FOR F=1 TO LEN A$
   40 LPRINT "■";
   50 NEXT F
   60 LPRINT TAB 3;"CONVERSION TA
BLE"
   70 LPRINT "       ▬▬▬▬▬▬▬▬▬▬▬"
   80 LPRINT
   90 LPRINT B$;TAB 10;"▮    ";C$
  110 LPRINT TAB 10;"▮"
  150 FOR E=1 TO SCA
  160 FOR F=Z(E) TO Y(E)+.00001 S
TEP X(E)
  160 LPRINT F;"..";TAB 10;"▮..";
TAB 14;INT (VAL D$*10**W(E)+.5)/
10**W(E)
  220 NEXT F
  230 NEXT E
  240 LPRINT "▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
  250 STOP
```

This will result in the complete list, with title, being printed on paper without stopping until the end. Should you require several copies of one conversion, do not reRUN after the first copy, just press the button on the printer to leave a short space, and type GOTO 20. This can be repeated to produce as many copies as desired.

Notes on listing

10 GOSUB start of program to get data.

20 Print title.

30–50 Underline title, making line right length.

60–90 More title and title to each column of list.

100 LH = point at which to start PRINTing AT.

150 For E = 1 to number of scales.

160 For F = beginning of scale to end of scale in required step.

170 Print dotted line for easy reading from value to converted value.

180 Print value, separating line and converted value.

200–210 Increase line to be PRINTed AT, and start new page if bottom of page has been reached – i.e., when program stops, pressing CONT continues at top of page.

220–230 Repeat until finished.

240 Final line showing end of list.

8000–8420 Input all factors relating to conversion.

These lines are fairly self-explanatory. The program will accept up to five scales. To have more, if necessary, simply add line 8385 CLS.

Listing

```
   1 REM CONVERSION
  10 GOSUB 8000
  20 PRINT A$
  30 FOR F=1 TO LEN A$
  40 PRINT "■";
  50 NEXT F
  60 PRINT AT 2,3;"CONVERSION TA
BLE"
  70 PRINT "          ■■■■■■■■■■■■■■■■■■■■■■"
  80 PRINT
  90 PRINT B$;TAB 10;"■    ";C$
 100 LET LH=7
```

INCHES TO CENTIMETERS

CONVERSION TABLE

INS.	CMS.
0	0
0.1	0.254
0.2	0.508
0.3	0.762
0.4	1.016
0.5	1.27
0.6	1.524
0.7	1.778
0.8	2.032
0.9	2.286
1	2.54
1.1	2.794
1.2	3.048
1.3	3.302
1.4	3.556
1.5	3.81
1.6	4.064
1.7	4.318
1.8	4.572
1.9	4.826
2	5.08
3	7.62
4	10.16
5	12.7
6	15.24
7	17.78
8	20.32
9	22.86
10	25.4
15	38.1
20	50.8
25	63.5
30	76.2
35	88.9
40	101.6
45	114.3
50	127

```
 110 PRINT TAB 10;"█"

 150 FOR E=1 TO SA
 160 FOR F=Z(E) TO Y(E)+.00001 S
TEP X(E)
 170 PRINT ".........."
 180 PRINT AT LH,0;F;" ";TAB 10;
"█..";TAB 14;INT (VAL D$*10**W(E
)+.5)/10**W(E)
 200 LET LH=LH+1
 210 IF LH>21 THEN LET LH=0
 220 NEXT F
 230 NEXT E
 240 PRINT "████████████████████
████"
 250 STOP

8000 PRINT "CONVERSION"
8010 PRINT
8020 PRINT " INPUT TITLE"
8030 INPUT A$
8040 PRINT A$
8050 PRINT "INPUT HEADING TO INI
TIAL VALUE  (8 LETTERS MAX.)"
8060 INPUT B$
8070 PRINT "INPUT HEADING TO CON
VERTED VALUE (8 LETTERS MAX.)"
8080 INPUT C$
8090 PRINT "INPUT CONVERSION FAC
TOR "
8100 PRINT "AS AN EQUATION IN F"
8110 PRINT "WHERE F IS THE VALUE
 IN ";B$
8120 INPUT D$
8130 CLS

8200 PRINT "INPUT HOW MANY SCALE
S"
8210 INPUT SA
8220 DIM Z(SA)
8230 DIM Y(SA)
8240 DIM X(SA)
8250 DIM W(SA)
8260 FOR F=1 TO SA
8270 PRINT "INPUT STARTING VALUE
 FOR SCALE ";F
8280 INPUT Z(F)
8300 PRINT "INPUT END VALUE FOR
SCALE ";F
8310 INPUT Y(F)
8350 PRINT "INPUT STEP"
8360 INPUT X(F)
8370 PRINT "INPUT DEC. PLACES"
```

```
6380  INPUT W(F)
6390  NEXT F
6400  CLS
6420  RETURN
```

How to Make Your Cassette Collection

After you have used your ZX81 for a while, you will have a number of programs stored on cassette. Some will be bought cassettes, others perhaps your own programs or programs copied from magazines or books. At some point the collection will have to be sorted.

Some people say that programs should be saved separately on C12 cassettes, one program per cassette, recorded three times on one side only. This is a costly method and one which occupies a lot of space.

I keep my collection mainly on C60 cassettes. There are programs recorded on one side only and a five-second gap is left between programs. The cassette recorder has a counter, and the count for the beginning and end of each program is noted on a card in the cassette case. When I want to load a program, I reset the counter, press fast-forward on the tape recorder until at the right place on the cassette and start loading. I do not record on the second side of the cassette. A C60 cassette is preferable, as it takes several minutes to reach the end of a C90 cassette even on the fast-forward winding. I usually fit about 6–12 short programs on a C60, or four long programs.

When starting with a ZX81 the tendency seems to be to save programs haphazardly – cassette 1 containing the first twelve or so programs, cassette 2 the next group, and so on. A program from cassette 1 may be updated, made better and then saved on cassette 3, and an even better version on cassette 4. If you have problems with LOADing, some programs may be saved several times in a row. This kind of system will eventually be rather cumbersome. It will be difficult to keep track of which programs are on particular tapes, how many programs you have in all and which are complete.

I faced this problem when I had filled six C60 cassettes and I then changed my system to the present one, which I have found very good and a pleasure to use.

I have separated my collection into programs which I wrote and programs done by other people (this was necessary because of sheer volume of programs, and is not really needed for a small collection).

The programs are separated into groups, each group having its own C60 cassette. There is one for Moving Graphics games, one for Static Graphics games, one for Word games, one for Educational programs, Practical programs, Utility programs such as line re-number, and so on. I specialize in Adventure games, so I have two tapes just for these. When one tape overflows, say Word Games, I just start a new cassette labelled 'Word Games 2'.

For the program or programs being worked on at the moment, it is necessary to have Current Work cassettes, which have to be numbered. Associated with these are sheets of paper in a file where notes of variables, ideas and so on are kept. The last cassettes needed are 'Security Tapes'. These are to minimize the possibility of a program being lost. If a cassette is lent to a friend who never returns it, or you mislay a cassette or accidentally wipe it, you do not want to lose a set of programs for ever. For this reason, the Security Tapes contain a copy of each finished program, not necessarily in any order, but with a note listing the programs. Since these tapes are not for everyday use, but only as a back-up system, convenience of loading can be sacrificed. Thus I use C90s or even C120s, recorded on both sides. The list of counts for each program on the cassette is on a card in the cassette case as usual. These should be stored in a different place from the 'everyday use' cassettes.

For convenience, a couple of special tapes can be made, for instance, of '20 favourite games' or 'programs to take on holiday'.

Finally, if you have a ZX Printer, it is useful to make a listing of each program which you have written yourself. This is not only another back-up system against accidental loss of programs, but also can be useful when working up such

programs into a longer or more complex one. These listings can be stuck on to a sheet of paper and put into a file, serving as future references.

Hardware Problems

Some of us are lucky enough to have a perfect ZX81, with a rock-steady RAMpack, an obedient ZX Printer, and which LOADs and SAVEs perfectly every time. However, most of us experience some sort of problem at one time or another.

It is not good for the system to dismantle it every evening to put into its box. The less you pull the RAMpack and Printer on and off the ZX81, the better. But it is not good to leave the system out to accumulate dust, either.

The way I solved this problem is by having the whole system on a board. Half-inch strips of plywood are used to make 'sockets' in which the ZX81 and ZX Printer sit, and the RAMpack is firmly seated on a wooden pedestal, so it cannot wobble. There is enough room to fit on a small cassette recorder, and the whole board has sponge non-slip 'feet' so that it will not scratch the table. The whole board can be moved to a storage shelf and has a plastic or wooden cover. If a special wooden cover is made with firm sponge pieces, glued in the right positions so that the whole box holds everything safely in place, the system can be stored on its side.

There are now large, tough keyboards available which can hold the ZX81 and the RAMpack too. This is ideal for those who dislike the touch-sensitive keys on the ZX81. However, if you are planning to obtain a printer, check to see whether this will be accommodated easily.

The Screen

Very few lasting problems are due to the TV set exclusively. Note that a black and white TV usually gives a better picture

than a colour one. Some large screens may lose synchronization when presented with moving graphics, while a smaller, portable TV, such as a 9-inch screen, always keeps the pictures stable.

The ZX81

To keep your ZX81 in good condition, always press the keys gently, using a stroking movement.

When a program gives instructions on which keys to use, make the instructions read 'PRESS NEWLINE TO . . .' rather than the commonly seen 'HIT NEWLINE TO . . .'

For those with new ROM ZX80s, problems may occur with the keyboard overlay if it is insecurely fitted. Make sure you fit it well and use with care.

The RAMpack

There are several RAMpacks available apart from the Sinclair 16K RAMpack. These vary in standard and performance. If you have a RAM expansion which does not have some sort of plastic casing, as the Sinclair 16K has, make a cover to protect your RAM.

When the RAMpack is attached to the ZX81, it should be gently pushed on, not rammed as hard as you can. If it won't work, try wriggling it slightly or pulling it slightly away from the ZX81, bringing the contact out by a millimetre or so.

Make adjustments ONLY while the power lead is pulled out.

If the RAMpack is not flush with the surface of the ZX81, and hangs rather precariously, cut a piece of sponge or balsawood as a support to hold it horizontally at the correct height. A ZX81 with a properly secure RAMpack should not even cause a flicker on the TV screen if the table is knocked.

Remember, when planning the layout of the board or casing to hold the ZX81 and the RAMpack with its support, that if

subsequently you get a printer, this will be attached between the ZX81 and RAMpack.

The Printer

This is attached by a double edge connector between the RAMpack and ZX81. The first roll of printing paper has a plastic centre tube, but refills have cardboard centre tubes, which may be slightly shorter than the original and slightly larger in diameter. The original hubs, which hold the roll of paper firmly, may fit more loosely into the cardboard tubes of the refills. In this case, wedge the hubs in with a little piece of cloth tape.

Problems with LOADing and SAVEing

This is probably the most frequent problem with the ZX81. There are several reasons why LOADing or SAVEing may be unsuccessful, and there are ways of coping with most of these fairly easily.

The cassette recorder should receive the program from the ZX81 at a constant speed, sufficient volume, and record it evenly, then be able to LOAD the program back. A fault in this chain will lose the program.

(1) The cassette recorder may run at an uneven speed caused by wear or poor design. The solution to this is to get another recorder.

(2) The recorder head may be out of alignment and, although a program can be successfully SAVED and re-LOADed from that machine, it will not LOAD on other, correctly aligned recorders. This can be cured by having the recorder head re-aligned professionally.

(3) The impedance of the recorder may be non-standard. Advice from an electronic shop or knowledgeable person can

usually solve this problem by adding or by-passing a resistor or tapping the loudspeaker connections directly.

(4) The inherent volume output can be very low, e.g., designed only for earphones. This could be overcome by adding an additional amplifier available from electronic stores.

(5) The recorder may have additional refinements for musical quality, such as Dolby, Limiter, Auto-record, Tone, Tune, Tape bias limiter and other settings. Switching off most of these, with Tone on High and experimenting, usually finds a compatibility with the computer. Otherwise, as previously mentioned, by-passing or tapping a connection will give good results. On hi-fi machines, the MIC and EAR sockets give better results with the computer than the line sockets.

(6) With stereo recorders some models, e.g., Sony 156, work better on the left channel. Often wiring a Y parallel lead into left and right channels for recording, and a single stereo headphone paralleled lead to the mono socket, will give better results. A low-impedance aerial lead often improves matters.

(7) Usually all leads for LOADing and recording can be left in but on some recorders it may be necessary to unplug the lead not in use at the time. (Note that the Spectrum must always have either LOAD or SAVE leads disconnected at one end.) This is usually to avoid an earth loop. Switches for this purpose can also be obtained. Full saturation without over-recording is the rule. A too low or too high level will lose the program.

Intermittent mains interference, e.g., from a fridge, electric typewriter or heater thermostat, can record a spurious signal which may result in loss of program both during LOADing and SAVEing and this type of interference is worth checking. The recording can sometimes pick up interference from the oscillator if the computer is too close to the recorder or the connecting wires are crossed or coiled. This is very infrequent, but it can happen.

(8) After prolonged storage the cassette sometimes replays unevenly and will not LOAD a program. Fast rewinding a few times, back and forth, will probably cure the problem.

(9) Occasionally, clean your recorder heads with a little spirit

on a cotton bud and demagnetize the head, otherwise a build-up of background noise can infringe on to the program.

(10) If you find that you still cannot LOAD correctly, there is a special cassette recorder marketed exclusively for the ZX81 (which is unsuitable for music). Although it is rather expensive for the quality of machine, it does give impressive results.

(11) If the problems persist with LOAD and SAVE, try the following. Devote a cassette exclusively to the program being dealt with. Leave the SAVEing leads plugged in to the computer and cassette machine. After every 20 lines, or every 15 minutes, SAVE the program so far. This way you will build up a series of recordings, getting gradually more complete. If the computer suddenly blanks then you can LOAD the last recording and you will only have lost, at most, 15 minutes' work. If that recording won't LOAD, try the previous one. Even if you have to go back three or four recordings, the chances are that you will not have lost all the program. At the end of a session, when you want to pack up, to continue next day or let the computer cool off, SAVE the program three times.

You may find that a cassette will mis-SAVE over a particular spot on the tape. This could be due to slight damage on the tape at that point or a previous signal being imprinted too 'deeply' for removal by wiping the cassette. In this case, just SAVE after the break.

In general, take more care when SAVEing than when LOADing. You can always re-LOAD, but if a program is mis-SAVED it could be lost for ever.

Glossary

The glossary covers most of the computer terms used in this book together with a few others which you may find useful. Note that some of the meanings given apply only to this book or to Sinclair computers and would not be suitable elsewhere.

Address   Units of memory are numbers between 0 and 255 called bytes, and each byte has an address. To find the value of a byte, you PEEK at its address, and to change it, you POKE to its address (if it is in RAM).

Amplification   The Spectrum does not produce very loud sounds and these can be amplified by plugging in the SAVE leads only and pressing Record on the cassette recorder without pressing Forward. Adjust the volume control to obtain the sound required.

Argument   The numbers or strings which you perform a function on constitute the argument of that function: e.g., for LEN A\$, A\$ is the argument, and for INT(X + Y/2), X + Y/2 is the argument of the function.

Arithmetic expression   With Sinclair Basic, commands can generally refer to expressions, as well as simple numbers: e.g., FOR F = VAL A\$(46) TO K+9/PI is possible in Sinclair Basic, but not in many others. An arithmetic expression is a collection of functions and arguments which the computer can reduce to a single number.

Array   An array is a set of variables with a dimension. DIM A(10,10,10) sets up 1000 variables, all initially zero, A(1,1,1) to A(10,10,10). String arrays can be made similarly: e.g., A\$(10,10). By omitting the last dimension, a whole string results, e.g., A\$(1) gives a 10-character string.

Attributes On the Spectrum each character square on the screen has attributes: an ink colour, a paper colour, and it can be extra bright or flashing. The attributes can be examined using ATTR.

Binary Binary is base 2. It uses ones and zeros only and is useful when defining user-defined graphics on the Spectrum. Each number from 0 to 255 can be represented by 8 binary digits, e.g., 36 is 100100 in binary.

Block graphics This refers to the graphics made from dividing a character square into four quarters and filling one or more of these with black or grey (or the INK colour on the Spectrum). They are used for low-resolution pictures and can be highly effective in patterns, bar graphs and other displays.

BREAK By pressing the BREAK key on the ZX81, or CAPS SHIFT BREAK on the Spectrum, a program can usually be halted. CONT will start the program again, but will clear the screen on a ZX81.

Bug Hardware or software which does not work properly contains bugs. These may be serious, e.g., clearing the memory when a key is pressed, or trivial, e.g., printing an extra full stop occasionally. A program often takes more time to debug than to write in its original form.

Byte Each unit of memory is called a byte and can contain a number between 0 and 255. A kilobyte is 1024 bytes, so 16K RAM contains 16384 bytes.

Character A character is a shape which the computer can produce which occupies one space on a 32×22 grid of squares. Numbers and letters are characters, and so are signs such as commas, full stops, dollar signs and so on. On the Spectrum you can make your own characters.

Character square A character can occupy any position on a grid of 32×22 positions on the screen. It must occupy an exact position, it cannot be half in one square, half in another. Each square on the grid is known as a character square.

Command Any keyword which actually changes or does something is a command. So GOTO, PRINT and LET are commands. SIN, for example, is not a command, as it only evaluates a number, but does not do anything with it.

Condition Use of conditions is the essence of computing. IF a condition is met, THEN do something. A condition can be true or false, there is no intermediate stage. A true condition has a value 1, a false one, 0. So LET A = A+(INKEY\$ = '8') will add 1 to A if the 8 key is pressed, but otherwise leave A as it was before.

Co-ordinate Any position on the screen can be specified by two co-ordinates. For characters, the first co-ord is the Y co-ord, with zero at the top of the screen, and the second is the X co-ord, with zero to the left. For plotted points, the X co-ord is first (zero to left) and then the Y co-ord (zero at bottom of screen). Print co-ords are the same on ZX81 and Spectrum, plot co-ords are different.

Core program The central routine of a program, where all other routines eventually go back to, is the core program. The term also refers to the simplest form of a program, to which other parts are gradually added.

CPU Central processing unit. On the ZX81 and Spectrum it is the Z80A, one of the best CPUs around.

Cursor The inverse character which you move along program lines when typing them in or editing them. The cursor controls, keys 5, 6, 7 and 8, have arrows on them, and are used as control keys in many games because of this.

Data list It is usually convenient to put all the DATA statements at the end of a program in a data list. On the ZX81 and the Spectrum much information may be stored in string arrays, and this too is a sort of data list.

Degrees Generally, when dealing with geometry we use degrees, where there are 360 degrees in a circle. The computer does not use degrees, but radians. There are $2 \times PI$ radians in a circle.

Dimensions   When an array is set up, it is DIMensioned, e.g.,
    DIM A(10,10). The number of dimensions here is 2. DIM
    A$ (4,6,9,3) has 4 dimensions.

Display   This refers to the screen and in this book is generally
    confined to the easily accessible part, i.e., the top 22
    character rows. There are two more lines at the bottom, used
    for errors, etc., and these can be POKEd into, but it is rather
    inconvenient.

Element   A single number in an array or letter in a string array
    is known as an element.

False   If a condition is logically false, its value is zero.

FAST mode   On the ZX81 only, FAST mode is available.
    Here the computer works at approximately four times its
    normal speed, but cannot keep the screen going as well. This
    means that during calculations, the screen goes blank.

Flickering screen   When in FAST mode, a ZX81 has a flicker-
    ing screen. Using PAUSE, even in SLOW mode, also causes
    a slight flicker and I have therefore tried to avoid its use in
    programs. On the Spectrum, there is no flicker due to
    PAUSE.

Floating point   Floating point arithmetic means that the
    computer keeps separate the digits of a number from the
    position of the point, just like a normal calculator.

Flow chart   This is a convenient way of planning a program.
    You write out the main routines of a program in boxes, such
    as 'move man', 'move alien', 'see if they meet' and put arrows
    between them to show what happens when a routine is
    finished. This can often help to guide you when actually
    making the program and checks that there are no loose ends.

Graphic letter   This refers to the Spectrum letters A to U
    which are produced when one letter is typed while the
    machine is in graphics mode. They are used when making
    your own user-defined graphics, and actually change their
    appearance in the listing. For this reason, I have used CHR$
    followed by the code of the character wherever possible.

Graphics   This is the general term for all the characters used for picture making and includes user-defined graphics as well.

Grey characters   Some of the low-res graphics on the ZX81 contain grey patches -- actually a fine chessboard pattern of black and white which appears grey. On the Spectrum these characters are not available, but you can define your own.

Hardware   The actual computer, the printer and cassette machine all come under the heading hardware.

Hexadecimal   Base 16. After 9 in hexadecimal you count A, B, C, D, E, F, 10. 10 hex is 16 in base 10. Hexadecimal, or hex, is used widely for machine code, as any 8-bit binary number can be represented by 2 hex digits.

High-resolution graphics   High-res graphics on the Spectrum are concerned with the PLOT, DRAW and CIRCLE commands, as these can be used anywhere on the screen. User-defined graphics are not true high-res, as these have to occupy character squares: a UDG cannot be half in one square, half in another.

INVERSE   An inverse character is one where the foreground and background colours have been swapped. These are available by pressing SHIFT GRAPHICS and a key on the ZX81 and by using INVERSE VIDEO on the Spectrum.

Keywords   All the ZX81 and Spectrum commands can be found on, above or beneath the keys. You never have to type a command in letter by letter, although you may have to press three different keys to get a keyword. This may be a little awkward at first, especially to those changing frequently between ZX81 and Spectrum, but eventually you will find it easy and convenient.

Kilobytes   A kilobyte is 1024 bytes and is usually abbreviated to K. So a ZX81 with 1K RAM contains 1024 bytes of memory for programs. However, some of this is used for the screen, and more for variables, cutting down on available memory.

Listing You will probably learn more about computer programming from examining listings of programs than in any other way. It helps if the listing is clearly set out, with line numbers in steps of 10, and each section of the program starting at the next hundred or thousand. Program descriptions should be at the very beginning (if short) or at the very end of the program, along with initial variables and arrays. This leaves the main program clear and uncluttered and relatively easy to follow.

Low-resolution graphics This refers to graphics which occupy a quarter or more of a character square. On the ZX81 this includes PLOT and UNPLOT as well as the block graphics. On the Spectrum, PLOT uses high-res graphics and so the low-res graphics are confined to the character set and any which you may define.

Main loop routine In many programs there is a central routine to which the program always returns. This is called the main loop routine, as distinguished from sub-routines.

Mantissa The mantissa part of a number consists of the actual digits and the decimal point or exponent is separate. An 8-bit mantissa means that a number is accurate to 8 figures.

Memory There are two types of memory on the ZX81 and Spectrum. ROM (read only memory) stores the Sinclair Basic, 8K on the ZX81, 16K on the Spectrum. RAM (random access memory) is for your programs and is 1–16K on the ZX81 (larger memory available but not from Sinclair), and 16–48K on the ZX Spectrum.

Memory address Each byte in memory has its own address, and can be PEEKed at or, if in RAM, POKEd to. Addresses in the ROM are from 0 to 16384, and RAM starts directly after this.

Movement vectors When moving an object on screen it is often necessary to keep the object's last position after you move it to the next. To do this, we alter the velocities to the right and top of the screen rather than changing the co-ordinates directly. These velocities are called movement vectors and show direction and speed rather than position.

Nesting   FOR-NEXT loops can be placed one inside another, e.g.,
  FOR F = 1 TO 10: FOR G = 1 TO 10:
  NEXT G: NEXT F.
  Here the G loop is said to be nested inside the F loop.

Pixel   A pixel is a single dot of black (ink colour). When you PLOT, you are plotting a pixel. ZX81 pixels are low-res, Spectrum pixels are high-res.

Plot grid   We can think of the screen being made up of a grid of squares, each square being able to contain one plotted pixel. On the ZX81, this grid is 64 squares horizontally, 44 vertically, and on the Spectrum, it is 256 by 176 squares. Remember that with PLOT, the X co-ord comes first, the opposite to PRINT AT, and the Y co-ord is zero at the bottom of the screen, whereas it is at the top for PRINT AT.

Priority   All operations have a priority, and the operations are carried out in order, highest priority first. Functions have a high priority, higher than $*$ and /, which are higher than + and $-$. When using operations, if you wish an operation with a lower priority to be carried out first, then bracket it. For example, PRINT INT RND$*4$ will always give 0, as INT has a higher priority than $*$. INT RND is 0, and 0 $*$ 4 is 0. To get RND$*4$ to be found first, it must be bracketed, giving PRINT INT(RND$*4$).

Procrustean   Assignment to string arrays is procrustean. This means that a string in an array always has the same length. In array A$(20,5), A$(1) always has 5 characters. If you type LET A$(1) = "RHINOCEROSES", A$(1) will become "RHINO", as it can only hold 5 characters. If you type let A$(1) = "HI", A$(1) will contain "HI", followed by 3 spaces.

Pseudo random   RND is not truly random, but follows a fixed sequence of 65536 numbers. By using RAND (RANDOMIZE on Spectrum) you can get the computer to start at a number determined by how long the computer has been on, so giving an unpredictable RND function.

Radian The ZX computers use radians rather than degrees. (See radian measure, p. 270.)

RAM Random access memory. This is the memory which you use for your programs. However, it is also used by the computer to store information on the screen, and some variables which it needs. This means that when a program runs it uses more memory and you do not have all the RAM to yourself even without a program. The ZX81 uses about 0.7K RAM for the screen, leaving little memory on a 1K ZX81. A Spectrum uses about 6.7K for the screen (due to the colour and high-res) which means that a 16K Spectrum has less programming space than a 16K ZX81!

Real time Games that use real time are the types where, if you just sit back and think, something will happen to you. The computer has a systems variable FRAMES which actually counts fiftieths of a second. Real time games generally use INKEY$ in a loop rather than INPUT.

ROM Read only memory. The ROM contains the Sinclair Basic. The larger the ROM, in general, the more commands and functions are available on a computer.

Rounding off/down INT always rounds a number down, removing everything after the decimal point. To round off, use INT(Number+.5).

Scientific notation For larger numbers, it is convenient to give a number less than 10 multiplied by a power of 10. This is called scientific notation. 6.35×10^8 is easier to read than 635000000, and for larger numbers, such as 4.93×10^{27}, writing out all the zeros is a waste of time.

Scrolling The ZX81 has a keyword command SCROLL. The Spectrum does not, but you can get one by using POKE 23692,255: PRINT AT 21,0: PRINT.

Shift The ZX81 has one shift key (and function), the Spectrum has two. These are used to get many of the keywords.

Software This refers to all the programs and attendant literature for your computer. This book is software, and so are cassettes of programs.

Systems variables   These are variables used by the computer. Their names are mnemonics only and are not recognized by the computer. Some, such as FRAMES, can be useful to PEEK at for specific purposes.

True   If a statement is logically true it has a value 1.

Undefined variable   A variable which is not initially given a value is undefined and will cause the program to crash when it is used.

User-defined graphic   On the Spectrum, you can design your own shapes, letters and symbols on 8 by 8 grids. The ability to make user-defined graphics is one of the most useful Spectrum features.

Variables   Variables used to hold scores, screen co-ordinates and string variables containing words required in a program have to be set (defined) initially, and this is usually done at the end of the program listing. This means that early in the program there will be a line such as GOTO 8000 to set up these variables.

X co-ordinate   This refers to how far along left to right on the screen an object is. The X co-ordinate starts at 0 on the left, and ends at column 31 on the right for PRINTing, 63 for PLOTting on the ZX81, and 255 for PLOTting on the Spectrum.

Y co-ordinate   This refers to how far up the screen an object is for PLOTting. It runs from 0 at the bottom, to 43 on the top for ZX81 plotting, and 175 on the Spectrum. For PRINT AT it starts at the top of the screen at 0, and goes down to line 21 at the bottom. This is rather confusing. In programs I generally use PLOT X,Y but PRINT AT Y,X, to act as a reminder of which axis is which.

Radian measure

When we use trigonometry, in general we use degrees. There are 360 degrees in a circle.

The computer uses radians, and there are $2*PI$ radians in a circle. To convert from degrees to radians, use:

number of radians = number of degrees $*PI/180$

and to convert from radians to degrees, use:

number of degrees = number of radians $*180/PI$

so 180 degrees is PI radians, 90 degrees is PI/2 radians and so on. Remember that SIN, COS, TAN and their inverses all operate on radians, so if in a formula you use SIN 37, on the computer use $SIN(37 \times PI/180)$.

If in a program the user inputs degrees, and the final answer uses degrees, it is necessary at the beginning to change the inputted values to radians, and then convert back at the end.

On the Spectrum, DRAWing curves uses radians to get the angle of curve. So to draw a semicircle, use DRAW x,y, PI and to draw a 60-degree arc use DRAW x, y, PI/3.

Just as for degrees, a small circle is used as a superscript, e.g., $60°$; for radians a superscript c is used, e.g., 1.3^c is 1.3 radians or approximately 74½ degrees.

Logic

A logical expression is one using AND, or OR, or NOT, or one which uses $=$, $<$, or $>$.

If a statement is logically true, it has a value of 1, if logically false, a value of 0. This is very useful in programs. For instance, that is how a command like LET $X = X+(INKEY\$ = $ '8')$-(INKEY\$ = $ '5') $+(X<2)-(X>30)$ works. Each bracket contains a logical condition which can be either true or false. Say that $X = 3$. When the program comes to the line above,

if key 8 is pressed, (INKEY$ = '8') is logically true, and has a value of 1. X is thus increased by 1. If the key is not pressed (INKEY$ = '8') is logically false, and X is unchanged. Note the −(INKEY$ = '5') which decreases X by 1 if key 5 is pressed.

This sort of program line is much shorter than

```
IF INKEY$ = "8" THEN LET X = X+1
IF INKEY$ = "5" THEN LET X = X−1
IF X <2 THEN . . . and so on.
```

Condensing a program by using logic will speed it up quite appreciably, as the computer takes more time to carry out IF statements that simple logical commands.

IN command

In the Spectrum manual, there is a short chapter on IN and OUT. These commands will be useful for controlling hardware attached to the Spectrum. However, IN has one very useful software application -- we can use it to read the keyboard. Although INKEY$ is easier to use, it cannot register 2 keys being pressed at the same time. With IN, however, pressing 2 keys will give a different signal to pressing 1 key.

The keys are grouped in half-rows, each group containing 5 keys. When no key is being pressed, IN will give a value of 255 for that address. Pressing the key nearest the centre of the keyboard (e.g., key 5 in the top left half-row) will reduce this by one, the next key out will reduce it by 2, the next by 4, the next by 8, and the outermost will reduce the value by 16. Thus any combination of keys can be recognized. This can be very useful to add extra features to some games, where you can move and fire at the same time or press 2 movement keys to move in a direction between the two. Another idea is to produce 2 player games, one player versus the other. One player can use the bottom left half-row (CAPS SHIFT to V) and the other can use the top right half-row (0 to 6).

The 8 addresses that IN uses to read the keyboard are given at the end of the IN chapter in the Spectrum user's manual.

Some useful PEEKs and POKEs

The systems variables on the ZX81 and Spectrum contain some useful values which can be PEEKed at and in some cases POKEd to. Here are a few you may find that you need:

ZX81

PEEK(PEEK 16398+256*PEEK 16399). This gives the code of the character in the screen position last PRINTed at. If you wish to know what character is at a particular position, say (Y,X) use PRINT AT Y,X;.
PRINT CHR$ PEEK(PEEK 16398 +256*PEEK 16399). This is very useful to detect collisions in games.

PRINT PEEK 16404+256*PEEK 16405−16384. This gives you an idea of how much RAM is used up by a program. BREAKing a program and doing this PEEK will give how much memory is used altogether including the screen and variables, by the program.

PEEK 16436+256*PEEK 16437. The ZX81 version of FRAMES. You can POKE these addresses to some value and use them as a countdown, counting down 50 every second. However, as PAUSE uses this, its use may be limited. Also, have RAND (if used) *before* you POKE FRAMES, or you may end up with predetermined 'random' numbers!

PEEK 16438 and PEEK 16439. These give the X+Y co-ords of last PLOTted point. This can be useful to check if a PLOTted object goes through a particular point on the screen.

Spectrum

- POKE 23561,n This gives the time that a key must be held down before repeating in fiftieths of a second. Initially 35, this can be POKEd to 255, for virtually no repeat, or other convenient values.
- POKE 23562,n Delay in fiftieths of a second between successive repeats of a key. This starts at 5, but for editing programs, 1 is a more suitable value.
- POKE 23606,m: POKE 23607,n For extra UDGs, and your own customized character set, POKE this systems variable CHARS to somewhere in RAM. This is discussed in Extending User-defined Graphics (pp. 48–55).
- POKE 23609,n This variable PIP gives the keyboard click length. Increasing its value to about 35 will give a more audible beep.
- PEEK 23672, PEEK 23673, PEEK 23674 Spectrum FRAMES. Can be POKEd to any value. Runs for nearly four days before resetting. You can set up user-defined functions to PEEK at FRAMES conveniently.
- POKE 23692,255 Scroll counter. POKEing this to 255 during a program stops the computer asking 'scroll?' when there is no space on the screen.

Appendix

ZX81 and Spectrum report codes

Code
0 Successful completion of program/command.
1 NEXT without FOR. See if you have accidentally started the loop with a different name.
2 Undefined variable. Check that variable is necessary and not mis-typed. If variable needed, make sure the program defines the variable in a LET or DIM statement before that part of the program.
3 Subscript out of range. Check that array is DIMensioned big enough, and that the subscript is within the required range if it is chosen using a complicated function.
4 Not enough room in memory. Programs can be reduced by shortening or omitting rules, tidying up to remove spare GOTO's and changing numeric to string arrays. If short program inexplicably uses up 16K, check that arrays have not been dimensioned too big.
5 No room on screen. PRINTing AT a y value of greater than 21, or forgetting to use scrolling and CLS can lead to this. Adding SCROLL or equivalent Spectrum POKE, or using CLS will cure this.
6 Arithmetic overflow. Number greater than 10^{38} generated. Check program for arithmetical errors.
7 RETURN without GOSUB. Check that you have not used GOTO to reach a sub-routine, and that the RETURN should not be a GOTO.
8 ZX81: INPUT as a direct command. Spectrum: End of file. Used with microdrive.
9 STOP statement. Useful to halt program at key points to

locate problems. Use CONT (CONTINUE on Spectrum) to carry on from next line.

A Invalid argument. Typically SQR for negative numbers, ASN for numbers larger than one and similar.

B Integer out of range. Check for arithmetical errors in GOTOs, GOSUBs, POKEs, etc.

C Text of a string argument is invalid for VAL (and VAL$ on Spectrum). Check that string which is being VALued has no excess spaces, letters, etc.

D BREAK pressed (during SAVE, LOAD, etc. on Spectrum). Also (ZX81 only) STOP at beginning of INPUTted data.

E Spectrum only. Out of data. Check lengths of FOR-NEXT loops reading DATA, where you have RESTOREd and that you have enough DATA.

F Invalid file name SAVE using empty string (or on the Spectrum one longer than 10 characters).

From now on Spectrum only:

G No room for line in memory. Wipe out line, CLEAR, reduce program and try again.

H STOP in INPUTted data.

I FOR without NEXT. Incorrect FOR loop, e.g., FOR F = 1 TO 0 without NEXT. Change FOR loop.

J Invalid input/output device. For Microdrive, etc.

K Invalid colour. Check INK, PAPER, BRIGHT and similar statements, especially if they use expressions to give colours.

L BREAK pressed during program.

M RAMTOP no good. You have CLEARed at too large or (more likely) too small a number for RAMTOP. CLEAR to a larger number.

N Statement lost. Typically in direct commands when you type them in wrong.

O Invalid stream. Used with Microdrive, etc.

P FN without DEF. Make sure your function is called what you think it is. DEFine FuNctions before using them.

Q Parameter error. Check FN statements to make sure that there are the right number of arguments, and that they are of the right type.

R Tape-loading error. Check cassette for faults, and re-SAVE program on different cassette making sure that volume and other settings on the cassette recorder are correct.

Index

Index

BETTER PROGRAMMING FOR YOUR SPECTRUM AND ZX81

will help you develop great games and serious programs for your Spectrum or ZX81. There are over 40 programs, each explained in detail so you'll understand exactly how the programs work and how to apply the techniques to your own programs. PEEK and POKE, high-resolution graphics, colour and sound — they're all covered.

The book teaches you better programming techniques and will help you get the very best from your computer.

Outstanding programs include:

FAIRWAY — play your ZX81 on a graphically illustrated 18-hole golf course.

AMAZE — you build the maze, and the computer solves it! Machine intelligence in action.

HERO MAKER — a major **ADVENTURE** program in which you — a young warrior of noble birth — must prove your valour by battling with the elements, dangers and monsters of the caverns of power.

KNIGHT FIGHT — jostle in a medieval joust with mounted knights. If you are knocked off your steed, the fight continues on foot.

3-D MAZE — solve the maze of the century, with every twist and turn of your path shown in 'three dimensions'.